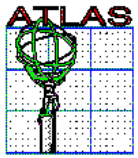


ATLAS
High-Level Triggers, DAQ and DCS

Technical Design Report

Issue: Pre-release Draft 2
Revision: 1
Reference: ATLAS TDR-xx
Created: 12 November 2002
Last modified: 19 March 2003
Prepared By: ATLAS HLT / DAQ / DCS Group

ATLAS Collaboration

CERN
European Laboratory for Particle Physics (CERN), Geneva

Acknowledgements

The authors would like to thank Mario Ruggier for preparing the template upon which this document is based and the DocSys group for their help in using it.

Table Of Contents

Part 1

Global View

1 Overview

This chapter introduces the overall organisation of the document into four parts and gives an overview of the principal system requirements and functions as well as listing the principal data types used in the system. A glossary is included at the end of the chapter for the reader's convenience.

The document has been organised into four parts:

- Part I - Global View

  Chapters 2, 3 and 4 address the principal system and experiment parameters which define the main requirements of the TDAQ system, the global system operations, and the physics requirements and event selection strategy respectively. Chapter 5 defines the overall architecture of the HLT/DAQ system and analyses the requirements of its principal components while chapters 6 and 7 address more specific fault tolerance and monitoring issues

- Part II - System Components

  This part describes in more detail the principal components and functions of the system. Chapter 8 addresses the design and performance of the dataflow component which is responsible for the transport of event data from the output of the detector readout links (ROLs) to mass storage as well as serving data to the High Level Trigger (HLT) system. Chapter 9 explains the composition of the HLT into a level 2 trigger and an event filter component and details the design of the dataflow within the HLT, the specificities of the HLT system supervision and the design and implementation of the event selection software framework which runs in the HLT. Chapter 10 addresses the online software which is responsible for the run control and supervision of the entire TDAQ system and the detector systems during data-taking. It is also responsible for many miscellaneous services such as error reporting, run parameter accessibility and histogramming and monitoring support. Chapter 11 describes the Detector Control System (DCS), responsible for the control and supervision of all the detector services such as gas and high voltage as well as monitoring many critical parameters of the detectors' operation. Chapter 12 draws together the various aspects of experimental control detailed in previous chapters and examines several use cases for the overall operation and control of the experiment, including: data-taking operations, calibration runs and required operations outside data-taking.

- Part III - System Performance

  Chapter 13 addresses the physics selection and performance. The common tools used for physics selection are described as well as the physics algorithms and their performance. Overall HLT output rates and sizes are also discussed. A first analysis of how ATLAS will handle the first year of running from the point of view of event selection assuming a specific machine startup scenario is presented. Chapter 14 discusses the overall performance of the HLT/DAQ system from various points of view, namely: the HLT performance as analysed in dedicated testbeds, the overall performance of the system in a testbed of ~10% ATLAS size, functional tests of the system in the detector testbeam environment. Data from these various testbeds is also used as input to detailed modelling of a full-scale ATLAS system.

- Part IV - Organization and Planning

  Chapter 15 discusses quality assurance issues and explains the software development process employed. Chapter 16 presents the system costing and staging scenarios. Chapter 17 presents the overall organisation of the project and general system resource issues. Chapter 18 presents the HLT/DAQ workplan for the next phase of the project up to LHC turn-on on 2007.

1.1 Main system requirements

This section will present the principal requirements on the HLT/DAQ system from several points of view.

1.1.1 From physics

Rejection from 75kHz to O(100 Hz)

Flexibility of selection (algorithmic, thresholds) - adaptability for new algorithmic strategies & for new (un-prepared for) physics signatures

1.1.2 From Read-out

Requirements & limitations from detectors in terms of readout bandwidth and also variation in LHC performance

1.1.3 Functional and operational

1.1.4 Long timescale operation

Requirements of modularity and the necessity of using commercial equipment

1.2 System functions

Describe briefly the principal system functions of the HLT/DAQ.

1.2.1 Detector Readout

1.2.2 Event selection and rate reduction

In particular, the introduction of two levels in HLT

1.2.3 Data transport

1.2.4 Experiment Operation and Control

1.2.5 Detector controls

The principal task of DCS is to enable the coherent and safe operation of the ATLAS detector. It supervises all hardware of the experimental set-up, not only the different subdetectors of ATLAS, but also the common experimental infrastructure. It also communicates with external systems like the infrastructure services of CERN and most notably with the LHC accelerator.

Safety aspects are treated by DCS only at the least severe level. This concerns mainly questions of sequencing operations or requiring conditions before executing commands. Also tools for interlocks both in hardware and in software are provided by DCS. Not in realm of DCS are situations, which could cause major damage to the detector or even endanger people's lives. The former is the responsibility of a dedicated Detector Safety System (DSS), with which DCS interacts, and the latter is addressed by the CERN-wide safety and alarm system.

It is mandatory that concerning the hardware of the detector all actions initiated by the operator and all errors, warnings and alarms are handled by DCS. It has to provide online status information to the level of detail required for global operation. Also the interaction of equipment experts with their subdetector should normally also go via DCS. DCS has to continuously monitor all operational parameters, signal any abnormal behaviour to the operator and give him guidance. It must also have the capability to automatically take appropriate actions if necessary and to bring the detector in a safe state.

Concerning the operation of the experiment, an intense interaction with the DAQ system is of prime importance. Good quality physics data requires detailed synchronisation between the DAQ system and DCS. Both systems are complementary in as far the DAQ deals with the data describing a physics event and DCS treats all data connected with the hardware of the detector. The former are organised by event number and the latter are normally categorised with a time stamp. The correlation between both is established in offline analysis.

Some parts of the detector will operate continuously because any interruption is costly in time or money or may even be detrimental to the performance of that detector. Hence its supervision by DCS is needed continuously. DAQ in contrast runs only when physics data are taken. Therefore DCS needs complete operational independence. This must however not result in boundaries, which limit functionality or performance. Therefore both share elements of a common software infrastructure. Different modes of operation are foreseen like taking data with colliding beams, detector calibration, and stand-alone operation of a subdetector or even of an individual detector element.

1.3 Types of data TDAQ deals with

Classification of various data types and associated requirements & limitations. Should also define specific terms here (more elaboration than in the glossary !!)

1.3.1 Detector control values

Orders of magnitude of amount of data, and its time variation frequency ... the need not to transport unchanged values

1.3.2 Event data

Byte stream ... what is it ... definition of event format

1.3.3 Configuration data

Define what constitutes configuration data, both for TDAQ and detectors in this context

1.3.4 Conditions data

What constitutes conditions data ... implications of overlap with offline

1.3.5 Statistics and monitoring data

Sources and sinks of histogram and general monitoring data ... what differentiates it

1.4 Long Term Perspectives

A more detailed view of the future planning is described in **Chapter 19**, but it is instructive to recall the principal elements here. In the current experiment installation schedule, the initial detector will be completed in December 2006, ready for the first LHC collisions in April 2007. A cosmic ray run is planned during the last few months of 2006. In its first year of operation, the LHC is expected to attain a luminosity of $2x10^{33}cm^{-2}s^{-1}$. The installation schedule of the TDAQ system is dictated both by constraints coming from the installation of the system itself as well as by the detector installation and commissioning schedule and in particular the needs of the detectors of TDAQ services during that time. The first elements of the TDAQ system will be required by the detectors in mid-2005.

This section should also discuss the fact that the final specification of various parts of the system will be required at differing times starting early 2004 and going on until mid 2006. The need to buy computing and network hardware as late as reasonable possible

1.5 Glossary

Is this really the place for the glossary or would it be better in the forward?

1.6 References

1-1

1-2

2 Parameters

This chapter is dedicated to the relevant parameters for the HLT / DAQ / DCS system. These include the detector readout parameters and the trigger selection for the correct dimensioning of the dataflow system and for understanding the data volumes that will need to be stored. These will be the subject of the first three sections.

Other important parameters for the correct definition of the system are the ones coming from the monitoring requirements. These are discussed in the fourth section.

The last section is dedicated to the DCS parameters: the subdivision of the system in detector parts and the amount of configuration data traffic in case of cold configuration and re-configuration of possible faulty elements.

2.1 Detector R/O parameters

This section could be moved to Section 1.2.1, "Detector Readout".

The ATLAS detector organized into three main systems: the Inner Detector, the Calorimetry and the Muon Spectrometer. These systems are then subdivided in sub-detectors.

The Inner Detector is divided in the following sub-detectors: Pixel, SCT and TRT. The Pixel sub-detectors is a detector using the pixel technology with a readout divided in ϕ regions and it is sub-divided in two endcaps, one inner barrel B-layer and 2 outer barrel layers. The SCT sub-detector is a Si microstrip detector subdivided into two endcaps and a barrel part subdivided in two regions for positive and negative η. The TRT sub-detector is a straw tubes tracking detector providing a particle identification based on the transition radiation.

The Calorimetry is a large system made of several sub-detectors based on different technologies. The barrel electromagnetic, the endcap electromagnetic, the endcap hadronic and the forward calorimeters use the LAr as sensible media with different absorbers depending on the particles to be detected. The barrel hadronic calorimeter and two endcaps at larger radii (with respect to the other calorimeters) in the range $|\eta| < 1.7$ is instead based on scintillator-iron technology: the Tilecal calorimeter.

The Muon spectrometer is subdivided in a barrel part where there are precision chambers based on Monitored Drift Tubes (MDTs) and trigger chambers based on Resistive Plate Chambers (RPCs). In the two endcaps up to $|\eta| \leq 2.4$, there are again MDTs as precision chambers and Thin Gap Chambers (TGCs) as trigger chambers. At large pseudo rapidities and close to the interaction point there are Cathode Strip Chambers (CSCs) that are suited to sustain the higher rate and the more severe background conditions.

In terms of readout signals to be transmitted to the Data Acquisition (DAQ) system, the LVL1 Trigger is another source of data and dedicated ReadOut Drivers (RODs) are used.

The organization in terms of readout is in fact slightly different from the pure division of the detector in sub-detectors and it is illustrated in the first sub-section, where a mapping of the ATLAS detector and trigger is specified in terms of data sources (the RODs) for the DAQ system in terms of the partitioning.

The concept of a partition used throughout this chapter coincides with the TTC partition concept introduced by the LVL1 TDR.

2.1.1 RODs per detector per partition

The distribution of the ROD modules and crates per sub-detector and partition generally follows the division of the sub-detectors in parts. This distribution assumes that there is no overlap of hardware among partitions and that each partition can be independently functional.

In Table 2-1 the number of RODs, ROD crates and ROLs are reported per sub-detector per partition.

Table 2-1 The distribution of the RODs per detector per partition.

	Detector	Partition	RODs	ROD crates	partitions	ROLs	Frag size (MB)
Inner Detector	Pixel		120	8	3	120	1.3
		B Layer	44	3			
		Disks	12	1			
		Layer 1 + 2	38+26	4			
	SCT		92	12	4	92	1.6
		Left Barrel	22	2			
		Right Barrel	22	2			
		Left Endcap	24	2			
		Right Endcap	24	2			
	TRT		256	22	4	256	1.0
		Barrel A	32	3			
		Barrel C	32	3			
		Endcap A	96	8			
		Endcap C	96	8			

Table 2-1 The distribution of the RODs per detector per partition.

Calorimetry	Tilecal		32	4	4	64	1.1
		Barrel A	8	1			
		Barrel C	8	1			
		Ext Barrel A	8	1			
		Ext Barrel C	8	1			
	LAr		192	16	6	768	1.4
		EMB A	56	4			
		EMB C	56	4			
		EMEC A	35	3			
		EMEC C	35	3			
		FCAL	4	1			
		HEC	6	1			
Muon Spectrometer	MDT		192	16	4	192	1.0
		Barrel A	48	4			
		Barrel C	48	4			
		Endcap A	48	4			
		Endcap C	48	4			
	CSC		32	2	2	32	0.2
		Endcap A	8+8	1			
		Endcap C	8+8	1			
LVL1 muon	RPC		32	16	2	32	1.0
		Half Barrel 1	16				
		Half Barrel 2	16				
	TGC		16	8	2	16	
		Endcap A	8				
		Endcap C	8				
	MIROD		1	1	1	1	0.104
LVL1 calo	CP / JEP	RoI	6	1 or 2		6	0.252
		CP	4		1	16	1.5
		JEP	4			16	1.1
		PP	4	8		16	
	CTP		1	1		1	0.012—0.038

These are the basic parameters updated during the 3rd ROD Workshop held in Annecy in November 2002. The fragment sizes reported in Table 2-1 have to considered as the maximum expected fragment size during the first phases of the ATLAS data taking and during the calibrations. A more accurate estimation of the fragment sizes is discussed in the next subsection (Section 2.1.2).

Concerning the LAr fragment size a more accurate estimation is on going in the community. It is based on both a compression of the data required and on zero-suppression schemes, but due to their nature it cannot be expressed in a straightforward way in a table like Table 2-1.

2.1.2 Fragment sizes per detector

Includes physics and calibration data.

Should have average values, spread and uncertainties; should be shown against luminosity; and against data compression schemes.

The fragment sizes reported in the previous table are indicative and they have to be seen as the maximum achievable figures.

Investigations are ongoing to obtain more realistic numbers for physics and calibration operations to resolve discrepancies with the values used in the Paper Model. The Detector people have to be contacted and an agreement on the numbers has to be found, based on the latest simulation they have for the sub-detector readout.

Table 2-2 Raw data fragment sizes in Bytes used for the paper model.

Subdetector	Low luminosity	Design luminosity
Pixels	200	500
SCT	330	1200
TRT	330	1200
E.m. calorimeter	752	752
Hadron calorimeter	752	752
Muon precision	800	800
Muon trigger (RPCs and TGCs)	380	380

For modelling the fragment sizes specified in Table 2-2 have been assumed (*see remark above table*). For the fragment sizes of the LVL1 RODs and the CSCs see Table 2-1.To each fragment a ROD header and trailer (together 48 Bytes) and a ROBIn header (56 Bytes) are added. A ROS subsystem in general will add several fragments together, after which a ROS header (52 Bytes) is added and a wrapper for the Data Collection software (36 Bytes). The network protocol will add further headers.

2.2 Trigger and Data Flow parameters

Two baseline “LVL1 trigger menus” are used in this report:

1. “low luminosity”: for a luminosity of 2.10^{33} $cm^{-2}s^{-1}$ with a LVL1 accept rate of about 20 kHz and a predominantly high transverse momentum trigger. The LVL2 accept rate is about 600 Hz,
2. “design luminosity”: for a luminosity of 10^{34} $cm^{-2}s^{-1}$ with a LVL1 accept rate of about 35 kHz. The LVL2 accept rate is about 1.5 kHz.

Editor: Note that the rates expressed above should be multiplied by a factor of 2 to 3 uncertainty to get the LVL1 input rates which the TDAQ system should be able to handle, namely 40 kHz at the low luminosity and 75 kHz at design luminosity.

The data needed for the LVL2 trigger and the type of processing performed by it depends on the regions of interest supplied by the first level trigger. Each of the four different types of RoIs (“muon”, “electron/gamma”, “jet” and “hadron”) has its own characteristic type of processing. The processing consists of several steps and after each step a decision is taken on whether data from other subdetectors within the region of interest should be requested for further analysis. The data rates can be estimated with the help of information on the type, rate, sizes and the locations of the regions of interest and on the mapping of the detector on the ROBins. See ch. ....for more details. In Table 2-3 and in Table 2-4 an overview is presented of the most important results obtained in this way for nominal LVL1 trigger rate. For estimating the size of the LVL2 farms and the number of SFIs the use of dual, resp. single 4 GHz PCs as compute servers has been assumed. The input bandwidth per SFI has been limited to 60 MByte/s.

The SFIs send the complete events for further analysis to the Event Filter. The Event Filter is expected to reduce the rate by a factor of 10 (see ch. ...) with a typical processing time of 1 second per event, which requires a farm of at least 300 or 750 dual-processor machines.

Table 2-3 Overview of important system parameters for nominal LVL1 trigger rate

	Low luminosity	Design luminosity
LVL1 trigger rate (kHz)	20.1	34.5
Average number of ROBIns receiving a RoI request, per LVL1 trigger	18.3	16.6
Average number of groups of 12 ROBIns receiving a RoI request, per LVL1 trigger	10.7	10.0
Maximum average output bandwidth per ROBIn (MByte/s)	1.5	2.8
Maximum average RoI request rate per ROBIn (kHz)	1.2	1.8
Maximum average output bandwidth per 12 ROBIns (MByte/s)	18.2	33.3
Maximum average RoI request rate per 12 ROBIns (kHz)	8.2	12.2
Total bandwidth LVL2 traffic (MByte/s)	325	532
LVL2 farm size	32	64
Fragment rate in = request rate out per L2PU (kHz)	6.8	5.4
Fragment volume in per L2PU (MByte/s)	10.2	8.3
Decision rate per L2PU (kHz)	0.63	0.54
Total bandwidth traffic to Event Builder (MByte/s)	731	2327
Event Building rate (kHz)	0.6	1.5
Number of SFIs required	12	39

Table 2-4 Overview of important system parameters for 75 kHzLVL1 trigger rate

	Low luminosity	Design luminosity
Average number of ROBIns receiving a RoI request, per LVL1 trigger	18.3	16.6
Average number of groups of 12 ROBIns receiving a RoI request, per LVL1 trigger	10.7	10.0
Maximum average output bandwidth per ROBIn (MByte/s)	5.7	6.1
Maximum average RoI request rate per ROBIn (kHz)	4.4	3.8
Maximum average output bandwidth per 12 ROBIns (MByte/s)	69.7	74.3
Maximum average RoI request rate per 12 ROBIns (kHz)	32.7	26.5
Total bandwidth LVL2 traffic (MByte/s)	1213	1156
LVL2 farm size	116	137
Fragment rate in = request rate out per L2PU (kHz)	7.0	5.5
Fragment volume in per L2PU (MByte/s)	10.5	8.4
Decision rate per L2PU (kHz)	0.65	0.55
Total bandwidth traffic to Event Builder (MByte/s)	2728	5060
Event Building rate (kHz)	2.2	3.3
Number of SFIs required	45	83

2.3 Monitoring requirements

Monitoring will require the transportation of monitoring elements (event fragments and/or monitoring results such as histograms) from the sources to the destinations. Investigations are under way to identify the latter ones and the traffic generated by the transportation. One may already identify :

some sources

- ROD/ROB
- ROS
- SFI
- Trigger processors (LVL1, LVL2, EF)

some destinations and the used networks

- private workstations + Control (CN) or Data Collection (DCN) network
- Online monitoring farm (possibly the EF Farm, or a sub-set) + Data Collection network

2.3.1 Monitoring matrix

The following table summarises our present knowledge of the relations between the sources and the destination for monitoring. We are still missing the figures for the expected traffic generated by monitoring activity. The table will be updated when feedback from the concerned groups is provided.

Table 2-5 Monitoring matrix

Source Destination	ROD/ROB	ROS	SFI	LVL1	LVL2	EF
private WS in Control Room	event fragments DCN	fragments DCN	events DCN			
private WS in Control Room	histograms CN			histograms CN	histograms CN	histograms CN
Online/EF Farm			events DCN	events DCN	events DCN	events DCN

2.4 DCS parameters

2.4.1 Data Volumes and rates

The DCS systems will be configured using ConfDB. PVSS (already mentioned and ‘defined’?) systems (DPTs, DPs, managers/drivers used etc.) will be configured and all associated software, such as hardware drivers(?), OPC servers, configuration files for ‘external’ applications (e.g. DDC) set up using information from ConfDB.

This ‘system’ level configuration will not be done very often, i.e. only at times of hardware modification (new equipment added, possibly equipment changed). Data volume large due to high number of separate systems (100 PCs?) though fast data rate not required as this is not a real time operation and should only be performed during shut-down periods.

During operation of the LHC, various operating modes are required for each sub-system. These modes require hardware to have different settings. A recipe is defined as a collection of settings used for a given operating mode. For a given run, more than one recipe may be required at different stages of the run (e.g. beam starting, beam on, beam stopping).

This ‘operating mode’ configuration is completed more often than the system level configuration. Before a run starts, any recipes used will be downloaded from the ConfDB and stored locally with each system (data consistency problem - may need to lock ConfDB at a given time before download). The data is then loaded into the running system and applied at the time it is

required. The data volume is lower than that for system configuration, though still quite large as there are many systems, each requiring a number of ‘stages’ containing all values to be set. Data rate required not high as the downloading is completed before a run starts (how long before?) and therefore, real-time download is not required.

Information held in ConfDB. System set up (system names in which PCs, manager lists, external application lists, configuration file(?) for external applications or at least information to allow these files to be produced, DDC, DPTs, DPs, addressing, dp functions, command transfer, message transfer). Recipe set up (original values, archiving, CondDB output configuration, alert limits, action scripts).

The CondDB will be used to store data read from DCS system. Bi-directional link desirable to allow data from both DCS and TDAQ to be correlated and displayed in PVSS. However, main direction is from DCS to CondDB. Desirable to put all values into CondDB (i.e. not only that ‘data required for off-line’). Data volume will be high (~1,000,000 channels plus others) though data rate could be low if data not sent in real time (possibly downloaded every 15mins / 1 h / 12 h / etc.).

2.5 References

2-1 3rd ROD workshop

2-2 Inner Detector TDR

2-3 LAr Calorimetry TDR

2-4 Tilecal TDR

2-5 Muon Spectrometer TDR

3 System Operations

Why this chapter? While Chapter 2, "Parameters", somehow says what is required from TDAQ in terms of performance, this chapter should highlight what is expected by TDAQ when it is "used".

The chapter contains the description of:

a. how an event is identified, at different levels in TDAQ.

b. What are the global TDAQ states. What are the detector and machine states. Including how TDAQ states relate to the detector and machine states. The global state transition

c. The definition of a run. How runs are identified. How an event is uniquely identified throughout the life of ATLAS. Types of runs. The question of the transition between runs. What is allowed during a run, what is done outside the run.

d. Partitioning: definition, operations

e. The general strategy to react to faults and errors (in TDAQ but also, and mainly, caused by external systems, such as the detector).

f. The role of data bases, what kind of data is permanently stored for what purpose (and where?).

3.1 TDAQ states

In the context of the global experiment, operations involve three main actors: the ATLAS detector (the detector for short in the following), the LHC machine and the TDAQ system. The high-level operation of the experiment, and the relationships between the main actors defined above, are described in terms of states. A state is a simple concept which summarises what a part of the experiment is capable to do at a certain point in time. States are also useful to describe how actors relate one to another.The further development of the concept of states, their refinement in terms of sub-states and their detailed relationships is the subject of CHAPTER 13

Three main states are useful to described the way TDAQ can operate:

- Initial: in this state the TDAQ system is not capable of performing any useful operation for the experiment. The only operations allowed on TDAQ are those which bring it to a situation where data taking can be performed (see below). This may be the state into which TDAQ is after e.g. a power failure, or TDAQ may revert to this state in order to re-initialise large parts of ATLAS.
- Standby: in this state the TDAQ system is ready, provided other conditions are met (for example related to the detector or the LHC machine), to initiate a data taking session. This means that the various parts and components have been properly initialised and set. For the purpose of detailing the initialisation procedures, the standby state may be reached by means of intermediate states: related for instance to loading software into processors, configuring components, etc.
- Running: in this state the TDAQ system is taking data from the detector.

For the purpose of the global operation of the experiment, two states are associated to the detector:

- Ready: the detector can be used for data taking.
- Not Ready (or $\overline{\text{Ready}}$): the detector cannot be used for data taking.

Three states may also summarise, for the purpose of operating TDAQ, the conditions of the LHC machine:

- Stable-Beams: the LHC machine is in a condition which allows safe operation of the detector and beams are available to produce physics events.
- Detector-Safe: the LHC machine is in a condition which allows safe operation of the detector.
- Non Stable-Beams ($\overline{\text{Stable-Beams}}$): the LHC machine is not in a condition which allow operating the ATLAS detector.

The inter-relationships of these states are indicated, for different types of data taking conditions, in the diagrams of Figure 3-1.

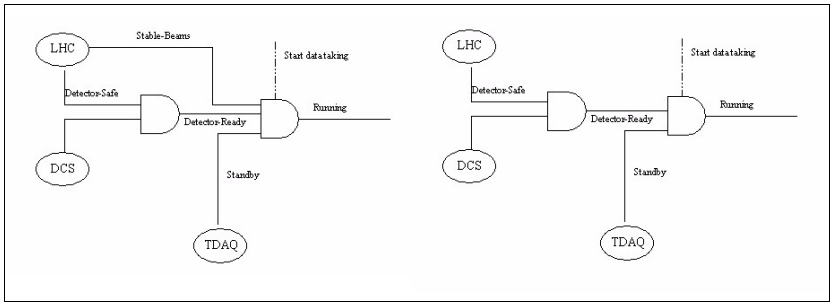


Figure 3-1 Inter-relationship of the detector, machine and TDAQ states.

3.2 The run

3.2.1 Run and Run Number

A run is defined as: a period of data taking in a TDAQ partition with a defined set of stable conditions related to quality of physics.*Note: define what are conditions*

The run number uniquely identifies a run (today it is a 32 bit number), it associates an event to a set of stable conditions. A run number is unique and it is never re-used during the lifetime of the experiment. A run number is generated by a central service.

The proposed mechanism to tag the run number into the event is based on the distribution of the run number (at the beginning of a run) to the ROD crate controllers. The RODs will then insert the run number into the fragment header for each event.

Any event fragment is identified, anywhere in the system, by its associated run number.

3.2.2 Event identification

Up to acceptance by level-2 (or event building in the case of a partition without level-2) an event is identified by an extended (32-bit) level-1 ID (generated by level-1 as 24-bit number and extended to 32).

A Global Event Number (GID) uniquely identifies an event, accepted by the level-2 trigger, within a run. It is generated by a central element (today it could be the DFM) after the LVL2 decision and it is made available, to be tagged into the event, to the element responsible for building the full event (today this would the SFI).

3.2.3 Requirements

The ATLAS TDAQ system is required to minimise its contribution to the experiment down time; although a quantitative definition of this requirement is not yet available, one can anticipate that the experiment down time due to TDAQ must be well below 1%.

There are two contributions to system down-time which are relevant to the subject of this document:

- the time spent by the system to initiate (start) or terminate (stop) a run, and
- the down time when coping with malfunctioning TDAQ components.

The two contributions above have an important impact on:

- how a run is defined, that is what configuration and parameter changes force a new run and what changes do not force a new run,
- how the transition between runs should be implemented, and
- how faults should be handled during a run.

It is difficult to identify precisely the conditions which characterise the quality of physics, hence a run. Nevertheless three typical classes of such conditions are: the parameters defining or affecting the selectivity of the triggers (LVL1, LVL2 and EF), the set of sub-detectors participating to the TDAQ partition, the operational parameters of sub-detectors. A modification of any of the above conditions forces a new run, that is the events following the change of the conditions are tagged with a new run number.

Conditions whose change forces a new run are stored in a conditions data base, whose contents are saved to permanent storage prior to the start of a new run.

Changes which do not force a new run include for example the removal or the insertion of processors or the disabling of FE channels (insofar as the physics is not affected)[1]. Those changes which do not force a new run are not stored in the conditions data base. The change may be entered for example in an electronic experiment logbook if it affects the performance of the TDAQ system (e.g. removal of an EF processor) or tagged into the event if it affects the data from the

1. The number of e.g. FE channels (or ROBs) which can be removed from the read-out without affecting the physics is bounded by some threshold which is sub-detector dependent. When the amount of unavailable read-out exceeds the threshold, the physics is affected and the run should be stopped.

detector. As an example of this latter: the removal of a detector or DAQ buffer may be flagged by appropriately setting a “quality flag” in the fragment header.

A run, when conditions do not change, may extend throughout an entire machine fill.

3.2.4 Categories of runs

Define what they are: what is their purpose, the actors (i.e. what a run type needs), where output goes.

Here we have:

physics run: (part of) the ATLAS detector, a fully functional (i.e. includes HLTs) TDAQ, DCS, the LHC machine, level-1 (including CTP).

calibration with beams: as physics run but sub-set of detector, no L2 or EF (but may reuse their infrastructure), LTP

calibration without beams: as above but no need for LHC machine (but make sure that detector can be switched on), LTP.

commissioning: as above.

3.2.5 Operations during a Run

There is the need for an active[1] run, i.e. following the successful execution of the run start command, to be interrupted temporarily: level-1 triggers are not generated so that some change or intervention on the detector can be done. Changes and interventions are such that they do not affect the physics, that is they do not force a new run. The global system state associated to the temporary interruption of a run is called Paused.

Two commands are available to respectively enter and exit the Paused state: pause and continue. Pause and continue commands may be issued: by an operator or by software (viz. an expert system).

When the pause command is issued:

- The level-1 triggers are blocked, by raising the global busy signal.
- All TDAQ elements are issued with the Pause command. Each element will execute it locally as soon as the handling of the current event is terminated (i.e. TDAQ elements will not empty their buffers before entering the paused state).
- TDAQ completes the transition to Paused as soon as all the TDAQ elements have entered the Paused state.

When the continue command is issued:

- All TDAQ elements are issued with the continue command, each element returns to the running state.

1. TDAQ is said to be in the running state.

- TDAQ completes the transition to the running state as soon as all the TDAQ elements have returned to the running state. At this point level-1 triggers are unblocked.

There is another special command which may be issued to a running TDAQ system, the Abort command. This command is reserved for very special cases and it entails a fast termination of the run; for example TDAQ elements will not complete the processing of events in their buffers.

3.2.6 Transition between Runs

From the operational point of view, a run is bracketed by a (run) start and a (run) stop command. These commands have to be sent to all[1] the TDAQ elements (viz. processors) for synchronous local execution prior to the transition to the running state.

Prior to the start of a run, or as the immediate consequence of a (run) stop command, the LVL1 Busy is asserted. The LVL1 Busy is removed upon the transition to the running state, this latter implies that all[1] TDAQ elements have completed their local execution of the (run) start command.

The completion of a run is also a process which needs synchronous local processing of all[1] the TDAQ elements: they receive the stop command, complete the processing of the contents of their buffers, produce end of run statistics etc. and leave the running state.

In addition to the run control command which signals a TDAQ element when a run is requested to complete, a mechanism is necessary to determine when the last fragment or event of the terminating run has been processed. This mechanism cannot be part of the run control, it is tightly related to the flow of the event data in the TDAQ system. This is done by means of a time-out: a TDAQ element will consider that the last event has been processed when 1) it has received the “stop run” command and 2) it has not received events for a certain time (for example a time out of some 10s of seconds).

The transition between two runs (i.e. stopping the previous and starting the next) includes two potentially time consuming processes:

- the completion of the processing of the contents of all the fragment / event buffers in the system: front-end buffers, RODs, ROBs, LVL2 and EF nodes;
- the synchronisation of all[1] the TDAQ elements to complete the transition stopped / running or running / stopped. That is, before the TDAQ partition may complete a state transition, all[1] the TDAQ elements have to have completed the transition locally.

There are conditions, for example the LVL1 trigger masks, thresholds and pre-scaling factors, or sub-detector calibration operating parameters, such that:

3. the modification of their values forces the change to a new run and
4. their value may be required to change relatively often (may be several times per machine fill).

1. It is envisageable that, in the case of the LVL2 and the EF, only a (to be defined) percentage of the farm needs to successfully perform the transition. The rest may do it “in the background” and join the new run afterwords.

The same considerations may also be applied to calibration runs, when some detector operating parameter may be required to change frequently.

In these cases the transition between runs as defined in Section 3.4 is not adequate in terms of the potentially long TDAQ system down time. A more efficient transition between runs is required and we define:

Checkpoint

> a transition in a running TDAQ system, triggered by a change in conditions or by an operator, which 1) results in the following events to be tagged with a new run number and 2) does not need the synchronisation, via run control start / stop commands, of all TDAQ elements.

The checkpoint transition is intended for those changes in conditions which require that events be correlated to the new conditions via a new run number but the change has a light implication on most of TDAQ. It is a mechanism to associate a new run number to events characterised by new conditions with minimal synchronisation within TDAQ.

A checkpoint transition is started automatically by the TDAQ control system when certain conditions are modified, it may also be initiated manually by an operator or automatically by some other software component (viz. an expert system). It should be noted that, for a transient time, events belonging to more than one run could be simultaneously present in the system. In particular given that LVL2 accepts are not time ordered, a EF node might have to process events belonging to two (or in principle even more) different runs.

The main feature of the checkpoint transition is the fact that events keep flowing in the system continuously: a mechanism is needed for a TDAQ element to detect when the new run begins. That is to say when the new run number becomes applicable, when the Global Event ID should be reset to 0 and when a TDAQ element should perform run completion processing and the initialisation necessary for a new run (for example a LVL2 processor may require to read the new conditions).

The run number may be used for this purpose, i.e. a TDAQ element recognises a new run whenever a piece of data (fragment or full event) is tagged with a new run number. TDAQ elements may therefore perform the “transition” from the old to the new run at their own pace and time. Note that the same mechanism is also applicable to analysis and monitoring software dealing with a statistical sample of the event data: an e.g. monitoring program recognises a new run whenever it samples an event with a new run number (with the caveat that, as for EF processing units, programs sampling events after Level-2 might have to handle events belonging to more than one run). A condition (belonging to a well defined sub-set of the possible run conditions) is changed or an operator asks for the execution of a checkpoint command.

3.3 Partitions and related operations

Definition of what a partition is: what for, who are the actors participating to the partition.

Allowed partitions (here we should make reference to the constraints imposed by the TTC system and include the table of the detector partitions).

What can be done with partitions: join and split.

Material from [3-2].

How partitioning is realised on the system is reserved to Chapter 5, "Architecture" (and possibly Part 2 System Components).

A list of the different ways the TDAQ system may be subdivided follows, each definition corresponds to a specific function.

- **TTC Partition**. A TTC partition includes a part of the TTC system and a corresponding part of the ROD-BUSY feedback tree. A TTC partition corresponds to a single TTCvi module. The concept of a TTC partition is already present in the Level-1 system.
- **TDAQ resource**. A TDAQ resource is the smallest part of the ATLAS TDAQ system which can be individually disabled (masked out of the ATLAS TDAQ), and possibly enabled, without stopping the data taking process.
- **TDAQ segment**. A TDAQ segment is defined as a smallest set of TDAQ system elements that can be configured and controlled[1] independently from the rest of the TDAQ system. A TDAQ segment may be also dynamically removed / inserted from / into an active TDAQ partition without stopping the run
- **TDAQ partition**. It is a sub-set of the ATLAS TDAQ system for the purpose of data taking.The full function of the ATLAS TDAQ is available to a sub-set of the ATLAS detector.

The word *smallest* separates the definitions above into independent concepts for the ATLAS TDAQ system[2]: resources can be disabled, segments can be removed and operated independently and partitions are fully functional TDAQ systems running on a sub-set of the ATLAS detector.

This is not an attempt to define a hierarchy within the ATLAS TDAQ. The concepts of (TDAQ) resource and segment are introduced because they are useful; the aim is that of formally and uniquely defining the different concepts which overloaded the term partition.

The term **partition** is reserved to the concept of a TDAQ partition. An equivalent formulation for a TDAQ partition is that the TDAQ system is required to be capable to run as multiple (fully functional[3]), possibly concurrent, instances each operating on sub-sets of the ATLAS detector.

1.That is the segment is capable of receiving commands from the TDAQ control system.

2.A segment cannot be a resource insofar as it is not the “smallest”, a resource cannot be a segment insofar as it cannot be operated. With the exception of the ROD crate, which may be seen both as a segment and a partition, a partition cannot be a segment since it is not the “smallest” and a segment cannot be a partition since, being the “smallest”, it cannot take data.

3.That is the complete functionality of the DAQ system is available to run a subset of the detector.

There exists one TDAQ Partition which covers the whole ATLAS TDAQ system. That TDAQ Partition is used for production data taking at the experiment. A TDAQ Partition always includes some FE elements of one or more sub-detectors in order to provide data. In one exceptional case, that of the DAQ partition, the TDAQ partition may include simulated input data instead of FE elements. A TDAQ Partition may stop at the level of one (or more) ROD crate(s) (ROD Crate DAQ), otherwise it will always include parts of the Event Building and parts of the Event Filter farm (i.e. it will always include a vertical slice of the DAQ system).

Operations on partitions.

3.4 Operations outside a run

Define what are the operations allowed when a run is stopped or when LHC is off. “Define” should include: the purpose of the operation, the actors, the expected result, the effect on TDAQ.

Initialisation, configuration.

Operations on partitions: split / join

A backup document with use cases would be useful (if can be done).

This section was moved with respect to the original layout so as to come AFTER the section on partitioning (since some of the operations might be done on partitions). It was also promoted one level up in the section hierarchy

3.5 Error/Fault reporting/handling strategy

Brief description of the global strategy here as the details are in Chapter 6. Emphasis should be given to 1) what TDAQ does when an internal error happens and 2) what TDAQ does when a fault happens outside TDAQ (but the fault affects the operation of the system).

The ATLAS TDAQ system will consist of a large number of, hardware and software, elements: a few thousand processors each running possibly several software processes. Today we do not have available a model for the MTTF of the full TDAQ or any of its components. However it is safe to assume that malfunctioning[1] in a system of such a size may happen at a non negligible rate. Under this assumptions and taking into account the requirement to maximise the TDAQ up-time we distinguish two types of behaviour:

1.Hardware fault (fan, power supply, disk, etc.) or software fault (process aborting).

- the fault happens in an element such that this latter can be removed from the running system without affecting the physics. It is the responsibility of the sub-system, to which the element belongs, to remove the element transparently without stopping the run. When necessary the sub-system will tag subsequent event with a “quality flag” value which marks events as “degraded”. Examples are: a ROD, a ROB (both require the tagging with a “quality flag”), a LVL2 or an EF processor or may be an entire farm (which do not require the quality flag).
- the fault happens in an element which is either essential (e.g. the DFM or the L2SV today) or such that it must respond to a high rate of requests (e.g. the ROS today). A fault in one of these elements is either fatal for the run or it may potentially generate a chaotic transient period (which may also be fatal to the run). The “simpler” fault tolerance defined in the first bullet above is not applicable. These TDAQ elements do have to be designed with a higher degree of reliability.

3.6 Data Bases

What has to be stored permanently? at least give some broad categories and the source of the data. Then list what is the required functionality of the data base system(s). For example data related to configuration, conditions, monitoring, etc.

Material from this section should come from the efforts going on to collect requirements on data bases.

3.7 References

3-1 GIWG. Run and States

3-2 GIWG. Partitioning

4 Physics selection strategy

This chapter provides an overview of the strategy for the online selection of events in ATLAS. The challenge faced at LHC is to reduce the interaction rate of about 1 GHz at the design luminosity of $1 \times 10^{34} cm^{-2} s^{-1}$ online by about 7 orders in magnitude to a rate of O(100 Hz) going to mass storage. Although the emphasis in this document will be on the contribution of the HLT to the reduction in rate, the final overall optimization of the selection procedure has to involve further tuning of the cuts at LVL1 as well.

The first section describes the requirements defined by the physics program intended to be studied by ATLAS, followed by a discussion of the approach taken for the selection at the HLT (and actually LVL1 as well). Next, a brief overview of the major selection signatures and their relation to the various components of ATLAS is given. Then, an overview of the various parts of the trigger menu for steady state running at an initial luminosity of $2 \times 10^{33} cm^{-2} s^{-1}$ is presented, together with a discussion of the expected physics coverage. This is followed by a description of how changes in the running conditions are going to be addressed, before finally ideas for the strategy of determining trigger efficiencies from data alone are presented.

4.1 Requirements

The ATLAS experiment has been designed to cover a large part of the physics potential expected for proton-proton collisions with a center-of-mass energy of 14 TeV at LHC. Amongst the primary goals are the understanding of the origin of the electroweak symmetry breaking (which might manifest in the observation of one or more Higgs bosons) and the search for new physics beyond the Standard Model, where for the latter it will be of utmost importance to be as open as possible to new processes, which are not yet modelled as of today. In addition, precision measurements of processes within (and outside of - if found) the Standard Model are to be made. These precision measurements will provide a very important consistency tests for any signals of new physics to be found. An overview of the variety of physics processes and the expected performance of ATLAS can be found in [4-1]. Most of the selection criteria used in the assessment of the physics potential of ATLAS are based on simple cuts, requiring mostly several high p_T objects, such as charged leptons, photons, jets (with or without b-tagging) and missing transverse energy.

The online event selection strategy has to define the proper criteria to efficiently cover the physics program foreseen for ATLAS, while at the same time provide the required reduction in event rate at the HLT. As discussed later, guidance on the online selection criteria has been obtained from the analysis cuts studied so far, reducing these even further to a simple set of very few signatures required for an event to be accepted.

The online selection at LHC is faced with a huge range in cross-section values for various processes, as shown in Figure 4-1. The interaction rate is dominated by the inelastic part of the total cross-section with a value of about 70 mb. The inclusive production of b-quarks occurs with a cross-section of about 0.6 mb, corresponding to a rate of about 6 MHz for design luminosity. It is worth noting that the cross-section for inclusive W production (including the branching ratio for the leptonic decays) leads at design luminosity to a rate of about 2 kHz. However the rate of some rare signals will be much smaller, e.g. the rate for the production of a Standard Model Higgs boson with a mass of 120 GeV for the (rare) decay mode into two photons will be below 10^{-3} Hz. The selection strategy has to ensure that such rare signals will not be missed while at the same time reducing the output rate of the HLT (to mass storage) to an acceptable value.

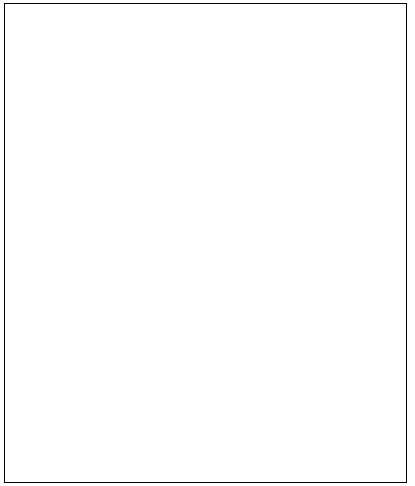


Figure 4-1 Cross-section and rates (for a luminosity of $1 \times 10^{34} cm^{-2} s^{-1}$) for various processes in proton-(anti)proton collisions, as a function of the center-of-mass energy.

Furthermore, the LHC environment at design luminosity implies the presence of about 20 inelastic events per bunch crossing, in addition to the possible event of interest.

The online selection thus has to provide a very efficient and unbiased selection, which should not spoil the excellent performance of the ATLAS detector. It should be extremely flexible and redundant to operate in the challenging environment of the LHC, providing also a very robust selection. It is highly desirable to reject fake events or background processes as early as possible in order to optimize the usage of the available resources. While presently the selection is based on rather simple criteria (however making use of the ATLAS capabilities to reject most of the fake signatures for a given selection), it is also thought to be indispensable to have additional tools for the online selection at our disposal.

4.2 The approach

In order to guarantee as best as possible openness to new physics, the approach taken is based on the emphasis of using inclusive criteria for the online selection, i.e. having signatures mostly based on single and di-object high-p_T triggers. Here high p_T refers to objects such as charged leptons with transverse momenta in the range of O(10 GeV). The choice of the thresholds has to be made such that a good overlap with the reach of the Tevatron (and other existing colliders) is guaranteed. Complementing this high p_T selection to enlarge the ATLAS physics potential requires to also have access to signatures involving more exclusive selections, using e.g. charged particles with transverse momenta of O(1 GeV) for the study of b-hadron physics.

The selection at the HLT will be based on the information already obtained at LVL1 and will exploit the complementary features of the LVL2 trigger and the EF selection. At LVL2, a fast rejection has to be achieved, using dedicated algorithms (to fulfil the latency constraints) needing

mostly only a few per cent of the event data (via guidance using the Region-of-Interest concept from LVL1). The selection signatures are to be refined at the EF, where the full event will be available for analysis, using more refined calibration and alignment parameters with less constraints on the latency. Furthermore, the EF will provide classification for the accepted events, in order to ease the offline analysis of the events.

Although the primary part of the selection will be based on mostly simple and inclusive signatures (to allow for future optimization), more refined selection tools have to be at the disposal. These include the usage of more refined algorithms (e.g. b-tagging of jets) or the application of more exclusive criteria, e.g. in order to enrich the available samples for certain physics processes. The chosen approach should be capable of supporting the ATLAS physics program over the full lifetime of the experiment.

Furthermore, it is highly desirable to select events with complementary and overlapping criteria, in order to avoid as much as possible biases due to the online selection.

4.3 Selection objects

The selection objects defined for the HLT can be based on the information of all sub-detectors of ATLAS, at the full available granularity. As mentioned above, the difference in definition of these objects between LVL2 and the EF will mostly refer to the complexity of the algorithm interpreting the raw data and the detail and level of accuracy for the alignment and calibration information used. In addition, the EF has the full event at disposal for the search for these objects.

ATLAS, as a multi-purpose detector, will have charged particle tracking (in the so called Inner Detector) coverage in the pseudo-rapidity region of $|\eta| < 2.5$ (inside a solenoidal field of 2 T) and fine-grained calorimeter coverage (esp. in the electromagnetic compartments) up to $|\eta| < 3.2$. The calorimeter coverage extends up to $|\eta| < 4.9$ for the measurement of missing transverse energy and forward jets. The coverage of the muon system extends up to $|\eta| < 2.4$ (for the trigger chambers) and up to $|\eta| < 2.7$ for the precision muon chambers. More details on the various components and their expected performance can be found in [4-1].

The following overview briefly summarizes the most important selection objects foreseen to be used at the HLT, where more details on the concrete implementation of the selection algorithms and their expected performance is to be found in CHAPTER 13.

- Electron: the selection of electrons will include a detailed shower shape analysis in the fine-grained calorimeters, a search for high p_T tracks and matching between the cluster and track(s) found. Further refinement is possible via the identification of Bremsstrahlung events and the application of isolation criteria.
- Photon: the selection of photons will be also based on a detailed calorimeter shower shape analysis, including the requirement of isolation, and possibly on the use of a veto against charged tracks (after conversions have been identified).
- Muon: the muon selection will make use of the stand-alone muon system to determine the muon momentum and its charge. A refinement of this information can be obtained by searching of tracks in the Inner Detector and matching of these candidates with the stand-alone muon track segment. Also for muons, isolation criteria can be required.

- Tau: the selection of taus (in the hadronic decay mode) will use the calorimeter shower shapes to identify narrow hadronic jets. These can be matched to one or more tracks found in the Inner Detector.
- Jet: the jet selection will based on calorimeter information, which might be refined by including the information from matching charged tracks as well.
- b-tagged jet: the selection will be based on an already defined jets, where the associated tracks found in the Inner Detector are used to search for e.g. large values of the impact parameter or for the presence of secondary vertices, as well as for soft (i.e. low p_T) leptons.
- E_T^{miss} : the definition of missing transverse energy will be based on the full calorimeter data, allowing for improvements via the inclusion of the information from charged particle tracks and from observed muons.
- total ΣE_T: again the calculation will be based on the full calorimeter information, with additional corrections being possible from charged tracks and reconstructed muons. A refined definition of the total ΣE_T can be made using only reconstructed jets.

In the following, a nomenclature of the type ‘NoXXi’ will be used, where ‘o’ indicates the type of the selection (‘e’ for electron, ‘γ’ for photon, ‘μ’ for muon, ‘τ’ for a τ hadron, ‘j’ for jet, ‘b’ for a b-tagged jet, ‘xE’ for missing transverse energy, ‘E’ for total transverse energy and ‘jE’ for the total transverse energy as obtained from jets only). The threshold on the transverse energy (or momentum) is given by ‘XX’ (in GeV). The required multiplicity of the object is indicated by the digit ‘N’ and the presence of an isolation requirement by the letter ‘i’. The thresholds listed refer to the true value, and the efficiency will depend on the actual implementation of the algorithm and the criteria applied, examples of which are given in CHAPTER 14.

As an example for the case of a more exclusive selection (requiring a more complex approach) is given by the case of b-hadron physics, where it is necessary to identify (semi-)exclusively the appropriate decay. This requires the identification of low p_T charged hadrons and leptons (electrons and muons) and might not always be able to rely on guidance to the HLT from LVL1. The starting point at LVL1 will be given by the presence of at least a low p_T muon.

4.4 Trigger menus

In this section, the present understanding of the trigger menu for a steady state running at an initial peak luminosity of $2 \times 10^{33} cm^{-2} s^{-1}$ is presented. Several parts of this trigger menu are distinguished and discussed separately:

- un-prescaled physics triggers, which form the backbone of the online selection and are chosen to guarantee the coverage of a very large fraction of the ATLAS physics program,
- pre-scaled physics triggers, which will extend the physics coverage for ATLAS (e.g. by having inclusive selections with lower threshold to enlarge the kinematic reach or by introducing more exclusive selections),
- dedicated monitor and calibration triggers (not already contained in one of the above items) to better understand the performance of the ATLAS detector, based on physics events not needed otherwise for physics measurements.

The description of these three parts of the presently foreseen trigger menu is followed by a discussion of the physics coverage achieved, including indications on how strongly the acceptance for selected physics processes depends on the actual choice of thresholds.

The derivation of the trigger menus and the actual values for the thresholds is following a strict procedure, starting from the physics analysis requirements, assessing the rejection capabilities at the various stages of the selection and at the end arriving at the estimates for the total HLT output bandwidth. This procedure has to be done obviously in several iterations, where also already existing information, e.g. from studies of the LVL1 trigger (see [4-3]) or from past studies of the HLT performance, as documented e.g. in [4-2], in taken into account.

Not discussed in this document are possible trigger selections dedicated to the study of forward physics or heavy ion interactions, as these additional aspects of the ATLAS physics potential are presently under investigation. It is expected that the excellent capabilities to identify high p_T signatures will also play a crucial role. Furthermore it should be noted that the menus are going to involve continuously, taking into account a more refined understanding of the detector capabilities and progress made in particle physics in the understanding of the Standard Model and the outcome of future searches for new physics before the start of the LHC.

A comprehensive assessment of the expected rates for the trigger menu will be given in CHAPTER 13.N, both for LVL1 (together with the corresponding trigger menu at this level) and for the HLT (i.e. rate to mass storage), including the expected total bandwidth of the accepted events to mass to storage.

4.4.1 Physics triggers

An overview of the major selection signatures needed to guarantee the physics coverage for the initial running at a peak luminosity of $2 \times 10^{33} cm^{-2} s^{-1}$ is shown in Table 4-1.

A large part of the physics program will rely heavily on the inclusive single (and di-) lepton triggers, involving electrons and muons. Besides selecting Standard Model events such as the production of W and Z bosons, gauge boson pair production, $t\bar{t}$ production (except the fully hadronic decay) and several decay modes of a Standard Model (and MSSM) Higgs boson(s), they also provide sensitivity to new heavy gauge bosons (W', Z'), to super-symmetric particles, large extra dimensions (via the Drell-Yan di-lepton spectrum) and to particle decays involving τ's (via their leptonic decay).

The inclusive and di-photon triggers will select a light Higgs boson via its decay H --> $\gamma\gamma$. The coverage for super-symmetry is extended by using the jet + missing transverse energy signatures as well as multi-jet selections, where the latter are especially relevant in case of R-parity violation. The inclusive and the di-jet trigger will be used in the search for new resonances decaying into two jets. Further sensitivity to super-symmetry at large values of tan β will be provided by signatures involving a hadronic τ selection.

Rare b-hadron decays and B-decays involving final states with a J/ψ are selected by a di-muon signature (requiring opposite charges) and additional mass cuts.

Table 4-1 Trigger menu, showing only un-prescaled physics triggers. The notation for the selection signatures is explained in Section 4.3.

Selection signature	Examples for physics coverage
e25i	W->lv, Z->ll, top production, H->WW(*) / ZZ(*), W',Z'
2e15i	Z->ll, H->WW(*) / ZZ(*)
μ20i	W->lv, Z->ll, top production, H->WW(*) / ZZ(*), W',Z'
2μ10	Z->ll, H->WW(*) / ZZ(*)
γ60i	direct photon production, H->$\gamma\gamma$
2γ20i	H->$\gamma\gamma$
j400	QCD, SUSY
2j350	QCD, SUSY, new resonances
3j165	QCD, SUSY
4j110	QCD, SUSY
τ60	charged Higgs
μ10+e15i	H->WW(*) / ZZ(*)
e20i+xEXX	W->ev
τ35+xE45	W->$\tau\nu$, Z->$\tau\tau$, SUSY at large tan β
j70+xE70	SUSY
xE200	
E1000	
jE1000	
2μ6 + $\mu^+\mu^-$ + mass cuts	rare B-decays (B ->$\mu\mu$X) and B->J/ψ (ψ')X

4.4.2 Pre-scaled and exclusive physics triggers

In Table 4-2 a prototype for additional contributions to the trigger menu in the form of pre-scaled physics triggers is given. These triggers are needed to extend the physics coverage of the online selection by extending on one hand the kinematic reach of various measurements towards smaller values e.g. of the transverse momentum in a process, and as well to include further selections by using more exclusive criteria.

A typical example for the application of these trigger selections is the measurement of the jet cross-section over the full kinematic range, starting from the lowest achievable E_T values up to the region covered by the un-prescaled inclusive jet trigger.

A second example is given by the extension of the selection for B-hadron physics to involve more decay modes. As discussed in more details already in [4-1] and [4-2], ATLAS offers the

Table 4-2 Examples for additional, pre-scaled or exclusive physics triggers

Selection signature	Physics motivation
single jets (8 thresholds between 20 and 400 GeV)	inclusive jet cross-section
di-jets (7 thresholds between 20 and 350 GeV)	
three jets (6 thresholds between 20 and 165 GeV)	
four jets (5 thresholds between 20 and 110 GeV)	
single electrons (4 thresholds between 5-25 GeV)	inclusive electron cross-section
di-electrons (2 thresholds between 5-15 GeV)	
single muons (4 thresholds between 5-20 GeV)	inclusive muon cross-section
di-muons (2 thresholds between 5-10 GeV)	
single photons (6 thresholds between 10-60 GeV)	inclusive photon cross-section
di-photons (2 thresholds between 10-20 GeV)	
taus (3 thresholds between 25 and 60 GeV)	
di-tau (thresholds TBD)	Z --> ττ selection
xE (5 thresholds between 45 and 200 GeV)	
E (3 thresholds between 400 and 1000 GeV)	
jE (3 thresholds between 400 and 1000 GeV)	
μ + γ (thresholds TBD)	
μ8 + B-decays	
b-jet (threshold TBD)	
filled bunch crossing random trigger	minimum bias events, trigger efficiency monitoring

possibility to perform several measurements of CP-violation in the b-hadron system. The selection strategy relies on selecting $b\bar{b}$ production via the semi-muonic decay of one of the b-quarks and then to semi-exclusively reconstruct selected decays, which involves the (possibly) unguided search for rather low p_T charged particles.

The pre-scale factors to be applied to most of the selections shown in Table 4-2 will have to be determined on the basis of the required statistical accuracy for the given application, taking into account the total available bandwidth. Furthermore, the ranges for the thresholds are to be seen as indicative only, the aim will be to cover the widest possible range, i.e. extending the coverage from the values of the nominal un-prescaled selection down to the lowest possible one for a given signature.

In the beginning of the data taking, it is essential that for all triggers the full detector will be readout to mass storage, in order to have the full spectrum of information available. It is however clearly envisaged, that a later stage some of the above pre-scaled triggers might no longer require the full detector information to be collected and thus more bandwidth could be made available for further selection criteria.

4.4.3 Monitor and calibration triggers

The selection signatures presented in Table 4-1 and Table 4-2 will already provide a huge sample of physics events, which will be of extreme importance for the understanding (and continuous) monitoring of the detector performance. At LHC, especially the leptonic decay of the copiously produced Z-bosons will be of great help, e.g. to determine the absolute energy scale for electrons and to intercalibrate the electro-magnetic parts of the calorimeter (using Z --> ee). The muonic decay will allow to set the absolute momentum scale both in the Inner Detector as well as in the muon system. In addition, the inclusive electron and muon triggers will select $t\bar{t}$ production (via the semi-leptonic decay of one of the top quarks). This sample will allow to determine the jet energy scale (via the hadronic decay to two jets of a W boson from the t decay) and to determine the b-tagging efficiency.

Table 4-3 Examples for specific monitor and calibration triggers, based on physics events, which are not covered in Table 4-1 and Table 4-2

Selection signature	Example for application
random trigger	zero bias trigger
unpaired / empty bunch crossing random trigger	background monitoring
e25 loose cuts	
e25 no isolation	
μ20 loose cuts	
μ20 no isolation	
γ60 loose cuts	
γ60 no isolation	
τ60 loose cuts	
2μ + Y mass	

Furthermore, samples of inclusive electrons will be used to understand the electromagnetic energy scale of the calorimeter, to perform alignment of the Inner Detector, to understand the energy-momentum matching between the Inner Detector and the calorimeter. Samples of inclusive muons are important as well for the Inner Detector alignment, to study their energy loss in the calorimeters and to understand / align the muon detectors. The calorimeter inter-calibration (especially for the hadronic part) will benefit from the use of inclusive (di-)jet samples.

However, there will be further requirements from the sub-detectors of ATLAS to collect samples of events for calibration, alignment and monitoring purposes. Some examples for such applications are:

- FILL IN HERE AND IN THE TABLE

These triggers give also explicit examples for selections which do not require always the full detector information to be read-out for mass storage. For some monitoring aspects, it could be even envisaged to not store any raw data, but just the result of processing the event at the EF (this would obviously only be possible after a stable operation of the detector and the machine has been reached).

4.4.4 Physics coverage

In the following, an overview of the major physics aspects foreseen by ATLAS will be given and the coverage achieved by the trigger menus as described above will be discussed. Special emphasis is given on describing the sensitivity to the actual value of the thresholds used in the selection signatures and to the redundancy achieved by the sets of signatures proposed.

As an example, the search for a Standard Model Higgs boson H in the decay mode to H -> $b\bar{b}$ will be discussed, where H is produced in association with a $t\bar{t}$ pair. The proposed selection criteria for this mode involve the requirement of a lepton from the semi-leptonic decay of one of the top quarks. In [4-1], the study has been based on the assumption of a p_T threshold for both the electron and the muon case of 20 GeV. The example in Table 4-4 shows the impact of raising one or both of these thresholds on the expected significance [4-4] for signal observation.

Table 4-4 Example of loss in significance for the associated production of $t\bar{t}H$

$p_T(e)$ >	20 GeV	25GeV	30 GeV	30 GeV	35 GeV
$p_T(\mu)$ >	20 GeV	20 GeV	20 GeV	40 GeV	25 GeV
$S/\sqrt{B}$	**1**	**0.98**	**0.96**	**0.92**	**0.92**

An overall optimization (which will be continued throughout the lifetime of the experiment) has to take into account the present understanding of physics, the effect of the foreseen thresholds on the acceptance for know (and unknown) physics processes, the rate for the selection (as the rejection can not be infinite due to e.g. the copious production of W bosons at LHC, which contribute via the leptonic decay to true electron and muon triggers), the inclusiveness of the selection (where one should note that a more exclusive selection does not necessarily imply a significantly reduced output rate -- as many final states will have to be considered in order to achieve good coverage) and the available resources.

4.5 Adaptation to changes in running conditions

Various effects will lead to changes in the running and operational conditions for the online selection, to which an efficient adaptation is needed. This is on one hand the drop of the luminosity during a fill of the LHC, on the other hand one has to foresee changes in the background conditions from the machine or changes in the detector performance, which will affect the selection. The strategy has to foresee as best as possible means to react to these changes and to keep the online selection robust.

4.5.1 Luminosity changes

During the life time of a machine fill of about 14 hours, the luminosity will drop by a factor of about 4. In order to profit from the available bandwidth in the HLT system, it should be foreseen to aim for a constant output rate of the HLT to mass storage. This could be achieved e.g. by including more trigger selections in the trigger menu, by adjusting pre-scale factors to fill up the available bandwidth with more selections using lower thresholds, with more exclusive selections or even by changing (i.e. lowering) the thresholds of the un-prescaled physics triggers. In this way, the physics coverage of ATLAS will be extended during a machine fill. One example is the case of B-hadron physics, where the un-prescaled trigger menu presented above is so far restricted to select only final states involving B-decays leading to at least two oppositely charged muons. As the luminosity drops below the peak value, other decay modes (e.g. fully hadronic ones or decays involving two low p_T electrons) can be added.

However one has to take into account that the sizeable change in the luminosity during a machine fill will imply changes in the average number of pile-up events present in the same bunch crossing, which might influence the optimal choice of threshold values (or isolation cuts) for various criteria in the definition of the selection objects and more studies are needed to assess whether simple changes of pre-scale factors are sufficient.

4.5.2 Background conditions

Furthermore, one will have to foresee adjustments to changes in the operating conditions, such as sudden increases in backgrounds or the appearance of hot channels, leading to certain triggers firing at a larger rate than acceptable. Furthermore, machine related background might increase suddenly and impact on the rate for certain selection signatures. Here, the first measure is most likely going to be either to completely disable this selection or to significantly increase the pre-scale factor.

4.5.3 Mechanisms for adaptation

From the operational point-of-view, it must be rather simple (and fast) to adjust the value for the pre-scale factors (and to include further trigger selections), changes to the threshold value might imply more complex re-initialisation procedures. Changes in the pre-scale factors could be done dynamically in an automated fashion (e.g. updating these number every N minutes), however it might be preferable to perform these changes less frequently in order to simplify the calculation of integrated luminosities for cross-section measurements.

The above list of changes in conditions is obviously incomplete and there will be many (as of today yet unknown) outside causes for changes to a stable operation, to which the online selection has to react and adapt, in order to preserve the physics coverage of ATLAS.

4.6 Determination of trigger efficiencies

An important aspect will be the capability to determine (as far as possible) the efficiencies for the above online selections from data alone. In this section, only a few basic ideas will be described, more quantitative details need to be worked out further. In addition, no explicit distinc-

tion between LVL1 and the HLT selections will be made, it is obvious that the efficiency determination will have to be done separately for each trigger (sub-)level. Furthermore, it is mandatory that the efficiency of the selection is monitored carefully throughout the lifetime of ATLAS, requiring that most of the lower threshold triggers needed for this are kept running all the time and that sufficient statistics is accumulated with these. In the following, an qualitative overview of various possible procedures is given. The importance will be on trying to determine the efficiencies in several ways, in order to minimize systematic uncertainties.

4.6.1 Bootstrap procedure

One possibility will be a bootstrap procedure, starting off with a selection of minimum bias events and using those to study the turn-on curve for very low E_T triggers on jets, electrons, photons, muons and so on. Next, the turn-on curves for e.g. electron triggers with higher E_T thresholds are studied using samples of events selected by an electron trigger with a lower (and thus already understood) threshold (and the same for all other object types).

4.6.2 Orthogonal selections

For the HLT, it will also be possible to foresee orthogonal (i.e. completely independent) selections in order to study the trigger efficiency of a particular step in the selection sequence, e.g. by using a sample which requires the presence of high p_T track in order to study the calorimeter (or the muon) trigger selections in more detail. Given the absence of a track trigger at LVL1, this will not be possible for the first stage of the selection. It could be envisaged however to use e.g. events selected by a muon trigger to study the calorimeter trigger response at LVL1 (using events where a jet is balancing a Z-boson, decaying to two muons).

4.6.3 Di-object selections

A further possibility will be the use of di-object selections (e.g. the plentiful produced Z --> ll decays) which have been selected online by a single object requirement and to study the trigger response for the second object (not required for the online selection). Here also other samples of resonances such as the J/ψ or the Y might prove very useful for the region of lower transverse momenta.

4.6.4 Required statistics

It cannot be emphasized enough that the amount of data needed to perform a proper understanding of the online selection is not going to be negligible in the start-up and throughout the lifetime of ATLAS, and will play an important role in assuring the potential of the experiment for discoveries and precision physics measurements. A more detailed assessment of the expected needs (before the start-up of ATLAS) should be done in the next years, presently a fraction of 10% is attributed to these triggers, which might not be sufficient and is clearly smaller than the ones presently used by running experiments.

4.7 Summary

An overview of the requirements and the basic principles for the online event selection strategy for ATLAS has been given. In order to maximise the potential of ATLAS for (presently) unforeseen physics, the online selection will follow an as inclusive approach, based mostly on high PT objects.

Details on the implementation of this strategy in terms of the software framework to perform the selection can be found in CHAPTER 9.N. In CHAPTER 13, more information on selection algorithm implementations and their performance in terms of signal efficiency and background rejection can be found. Finally, CHAPTER 14.N addresses the issue of system performance of the online selection, presenting the present understanding of the resources (e.g. CPU time, network bandwidth) needed to implement the selection strategy presented in this chapter.

4.8 References

4-1 ATLAS Detector and Physics Performance TDR.

4-2 ATLAS HLT, DAQ and DCS Technical Proposal.

4-3 ATLAS LVL1 TDR.

4-4 E. Richter-Was, private communication

5 Architecture

The purpose of the chapter is to describe the top level architecture of TDAQ, in terms of

its place with respect to the other parts of ATLAS, as well as systems and services external to ATLAS,

how the system is organised: functionally, in terms of sub-systems and in terms of more abstract elements,

a generic architecture with a a definition of the abstract components that are visible at the architectural level

how sub-systems map onto the generic architecture ("views").

how the scalability and partitioning can be performed and

finally it proposes a baseline architecture expressed by the realisation of the abstract components.

DCS is considered, as regards this chapter, as a black box with interfaces to TDAQ and external systems. The internals of DCS do not belong to this chapter.

5.1 TDAQ context

5.2 Context Diagram

The ATLAS TDAQ context diagram is shown in Figures 5-1. The LVL1 trigger provides LVL2 with region-of-interest (RoI) and other data needed to guide the LVL2-trigger data selection and processing; this interface is discussed in detail in part 2. The Timing, Trigger and Control (TTC) system provides signals associated with events that are selected by the LVL1 trigger. ReadOut Drivers (RODs), associated with the detectors, provide event fragments for all events that are selected by the LVL1 trigger. In addition, the LVL1 system contains RODs which provide data to be read out for the selected bunch crossings. The LVL1 trigger system, the TTC system and the ROD systems of the detectors all need to be configured by the DAQ system, for example at the start of each run. These components are shown in the top part of the diagram.

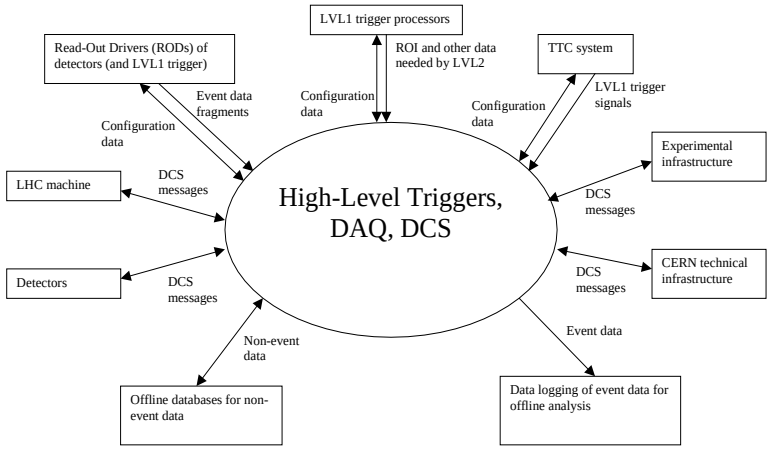


Figure 5-1

Interfaces to other external systems are also illustrated in Figure 5-1. These connect to the LHC machine (e.g. to exchange information on beam parameters), to the detectors (e.g. to control voltages), to the experimental infrastructure (e.g. to monitor temperatures of racks), and to the CERN technical infrastructure.

The remaining interfaces relate to long-term storage of data that must also be accessed for off-line analysis of the event data. For events that are retained by the high level triggers, the event data have to be stored for offline analysis. In addition, a large amount of non-event data has to be stored: alignment and calibration constants, configuration parameters, etc. Not shown in the figure is the importation of programs from the offline software for use by the high level triggers.

5.2.1 TDAQ Interfaces

The ATLAS TDAQ system interfaces to a variety of other subsystems inside ATLAS as well as external systems which are not under the experiments control. The following describes these interfaces in terms of

- Partners involved (TDAQ subsystem and non-TDAQ system)
- Responsibilities for the interface on both sides
- Data exchanged via the interface.
- Pointers to documentation on interfaces including data formats.

The interfaces are distinguished into two classes, those to other parts of ATLAS and those to external systems.

5.2.1.1 TDAQ interfaces to ATLAS

5.2.1.1.1 Level 1 Trigger

Although technically part of the ATLAS TDAQ, the Level 1 trigger is here seen as an external subsystem from the point of view of the DAQ and the HLT components.

The sole interface to the Level 1 trigger is provided through the RoIBuilder. It connects to both the CTP and the Calorimeter and Muon Level 1 triggers and receives direct input from them. This information is combined on a per event basis inside the RoIBuilder and passed to the Level 2 system.

The physical interface is provided by an SLink. There are inputs from nine different sources into the RoIBuilder. The interface has to run at full LVL1 speed, i.e. 75 kHz. Data flows from the Level 1 system to the RoIBuilder while the SLink interface provides only flow control information in the reverse direction. Asserting XOFF is the only way for the RoIBuilder to stop the trigger. The specification of the interface is described in Ref[L1L2Interface].

Only one partition at a time can include the Level 1 and Level 2 trigger. This will be the default trigger during a physics run.

5.2.1.1.2 Detector specific triggers

For testbeams, installation and commissioning as well as calibration runs it will be necessary to trigger the data acquisition for a subset of the full system independent from Level 1 and Level 2 and in parallel with other ongoing activities. These triggers are referred to as detector specific triggers. For event building a Data Flow Manager (DFM) per partition is assumed.

Any detector specific trigger will communicate via the TTC system with its corresponding DFM. The DFM component therefore requires a TTC input and a mode where it will work independent from Level 2. The details of how the trigger itself is implemented are left to the subdetectors. The DFM must be able to throttle the detector specific trigger via TTC.

5.2.1.1.3 Detector Front-ends

The detector front-ends provide the raw data for each event that Level 1 triggers on. The detector side of the interface is the Read Out Driver (ROD) while the TDAQ side is the Read Out Buffer (ROB). The connection between the two is the Read Out Link (ROL).

From the point of view of the detector side the interface follows the SLink specification. Implementation details can change as long as the specification is followed and most RODs provide room for a mezzanine card to hold the actual interface.

Data flows from the ROD to the ROBs, while only the SLink flow control is available in the reverse direction. This interface has to work at the full Level 1 accept rate, i.e. 75 kHz.

5.2.1.1.4 DCS

There is a two-way exchange of information foreseen between DCS and the rest of the TDAQ system. DCS will report information to TDAQ about the status and readiness of various components and TDAQ will both provide configuration information and issue commands related to runs. DCS is also the only interface to all information regarding the LHC machine.

All this communication happens via mechanisms defined and provided by the Online Software. ref[DCS-OSW communication packages].

5.2.1.1.5 Detector Monitoring (ROD Crate to Online SW)

5.2.1.1.6 Conditions Database

The conditions database will store all time-dependent status information of the system that is important for reconstructing events. Components of the HLT / DAQ system will mostly write information into the database, but some like the Event Filter will also read from it.

The concrete interface to the conditions database is not yet defined. Assuming that the implementation is making use of a relational database a variety of communication mechanisms will be available. It remains to be studied how accessing the conditions database will affect the HLT performance itself and how frequently this will occur.

5.2.1.2 External interfaces

5.2.1.2.1 Mass Storage

Events that have passed the Eventfilter will eventually be written to mass storage. It is assumed that this service itself is provided by the CERN computing division. In the current design the Sub Farm Output (SFO) component produces a series of raw data files which are stored on disk to provide local buffering if necessary (e.g. when the network connection to the CERN main site is down). A separate process picks these files up and transfers them to the computing division.

The data files are stored in a well-defined format and libraries are provided to read these files in

Table 5-1 Overview of interfaces between TDAQ and other ATLAS or external systems.

Interface	Data Rate	Data Volume	Data Type
LVL1 Trigger	75 kHz		Trigger and RoI data
Detector specific trigger	xx kHz		Trigger
Detector Front-ends	75 kHz	135 GByte/s	Raw data
Detector Monitoring	few Hz	few MByte/s	Raw data
DCS	few Hz	??	Control information
Conditions Database	??	??	System status
Mass Storage Interface	200 Hz	360 MByte/s	Raw data + LVL2 and EF results

offline applications.

5.2.1.2.2 LHC

All communication between the LHC machine and TDAQ is done via DCS. Therefore there is no direct interface between any other TDAQ component and the LHC. The communication mechanisms are described in the section on DCS.

5.3 TDAQ Organisation

The purpose of the section is to show how TDAQ is organised (the system as such, not necessarily managerially) internally. The internal organisation is looked at from three perspectives: what function are performed by TDAQ, how functions are associated to TDAQ blocks, and a very abstract categorisation of internal elements. Generality (as opposed to implementation) and complementarity of views is stressed.

5.3.1 Functional decomposition

The TDAQ system provides the ATLAS experiment with the capability of: moving the detector data (physics events) from the detector to mass storage, selecting, between detector and mass storage, those events which are considered of physical interest, controlling and monitoring the whole experiment.

The following functions are identified:

- Detector read-out: the data produced by one bunch crossing are stored in detector memories (RODs), an event is therefore split in a number of fragments: there are ~ 1600 of such memories which have to be read-out at a rate of 75KHz into a set of TDAQ buffers (the event memory for TDAQ).
- Movement of event data: once buffered event fragments have to be moved to the high level triggers and, for selected events, to mass storage. This is a complex process which involves both moving small amounts of data at the level-1 trigger rate (the region of interest data for the level-2 trigger at 75 KHz) and the full event (i.e. ~ 1MB) at the rate of the level-2 trigger (few KHz).
- Event selection: TDAQ is responsible to reduce the rate and the data volume to the manageable amount of ~ 100MB/sec; this is achieved by a sophisticated, 2-level, trigger system.
- Event storage: events selected by the high level trigger system are written onto permanent storage for further offline analysis.
- Controls and monitoring: this refers to the capability of i) operating and controlling the experiment (detector, infrastructure, TDAQ) and ii) monitoring the state and behaviour of the whole of ATLAS.

5.3.2 TDAQ building blocks and sub-systems

The ATLAS TDAQ system is designed to provide the above functions in terms of the following building blocks:

- Read-Out System (ROS): event data is buffered, by the ATLAS detectors, in the RODs; each ROD holding a fragment of the whole ATLAS event. The ROD fragments are read by TDAQ into its own buffers, the “Read-Out Buffers” (ROBs). Logically, but not necessarily implementation-wise, there is an equal number of ROB buffers as there are ROD fragments (indeed, see below, the level-2 trigger needs to access data at the level of the individual ROD fragments). Event fragments are kept in the ROB buffers until they are either moved downstream (accepted by the level-2 trigger) or they are removed from the system (rejected by level-2). The depth of the ROB buffers is determined by the time needed by level-2 to select events. The ROS provides individual event fragments, out of the ROBs, to the level-2 trigger and to the event builder: in this latter case a further level of buffering, multiplexing several individual ROBs into a single event builder input, may be provided by the ROS.
- Level-2 trigger: the level-2 trigger, as detailed in XXXXX, uses a mechanism to selectively read-out an event; that is, the level-2 trigger requests, as directed by the findings of the level-1 trigger, a small fraction of the event fragments in order to take a decision on the acceptance/rejection of the event. The ROI mechanism, using input from level-1, defines what fragments the level-2 trigger will need for a particular event. Appropriate fragments are requested from the ROBs and used to decide on the acceptance or rejection of that event. It is remarked that the level-2 trigger requests fragments on the basis of i) the level-1 identifier and ii) the ROL number (as opposed to a ROB number).
- Event Builder: the event is kept in the form of many (~1600) parallel streams up to the decision by the level-2 trigger. Any further reduction in the event rate needs working on the complete event, hence the requirement for a component which merges all the fragments of an event into a single place: the event builder.

- Event Filter (EF): another level of event rate reduction is provided by the event filter which requests complete events from the SFI buffers and performs on them complex selection algorithms.
- TDAQ controls: the function in charge of the control and supervision of the whole TDAQ system; this includes the initialisation and configuration of the TDAQ components. It also includes those ancillary functions, such as sharing (non event data) information between components.
- Detector controls: it represents the function in charge of controlling and monitoring all aspects of the ATLAS detector; it also includes the function of initialisation and configuration of the ATLAS detector.
- Monitoring: it is the part of TDAQ in charge of i) the (event data based) monitoring of the experiment and ii) the operational monitoring of TDAQ.

Now explain that the work has been organised in terms of a dataflow, dcs, hlt, pesa, online sub-systems

For the purpose of the organisation of the development, the work has been organised in terms of broad, function oriented, sub-systems:

- The dataflow sub-system: it is responsible for the development of the detector read-out and data transport functions.
- The high-level trigger (HLT) sub-system: it is responsible for the development of the level-2 and event filter components. It is in turn organised in terms of
  - HLT infrastructure: responsible for the development of the infrastructure necessary to support the trigger and filter algorithms, and
  - physics and event selection architecture (PESA): to achieve a coherent description of the physics needs which will drive the strategy for the selection of events in the ATLAS Trigger/DAQ system, with the most emphasis on the High Level Trigger.
- The online software sub-system: it is responsible for the development of all the supporting software, such as the one related to controls, data bases, monitoring, etc.
- Detector Control System (DCS): it comprises the control of the subdetectors and of the common infrastructure of the experiment and the communication with the services of CERN (cooling, ventilation, electricity distribution, safety etc.) and the LHC accelerator

5.3.3 Component categories

Here we characterise the components of TDAQ in terms of broad categories of elements: buffers, processors, supervisors and networks. It will be indicated what buffers (decouple parts of TDAQ, smooth differences in performances between parts of TDAQ), processors (selection at HLT level, monitoring, control), supervisors (L2SV, DFM) to control the flow of the data) and networks (transport the data) do in the system.

In very broad terms the ATLAS TDAQ system is composed of

- Buffers: they are used to decouple the different parts of the system: detector R/O, level-2, event builder and event filter. Because of the parallelism designed into the system, buffers belonging to the same function (e.g. ROBs) are independent.

- Processors: to run event selection algorithms, to monitor and control the system. They are organised in farms, groups of processors performing the same function.
- Supervisors: these elements coordinate the parallelism, in terms of assigning events to processors and buffers, at the different levels: the level-2 trigger (RoIB and L2SV), the event builder (DFM) and event filter.
- Communication systems: they connect buffers and processors to provide a path for transporting event data or a path to control and operate the overall system. Communication systems are present at different locations in the system, some of them are switching networks, others may be point to point links.

5.4 TDAQ generic architecture

Now we put together and refine what we have said in Section 5.3, "TDAQ Organisation". The generic architecture is built upon and justified on the basis of that section. A backup document may be needed if more detail is required.

5.4.1 Architectural components

This is the list of the components which are visible at the level of the architecture, the list should include what is relevant in terms of functions, building blocks and abstract elements.Should include the functions and components identified in Section 5.3, "TDAQ Organisation": ROD,..., RoIB, L2SV, online software major components such as control, DB, etc. For each component the following information should be provided: a definition or purpose (i.e. its function), and required performance. This section is neutral with respect to possible implementations. Why generic (and "unorthodox") names such as RRC and RRM, ROB instead of ROBin? The intent is to indicate that at that point in the system something is needed with a certain functionality (to connect and possibly mpx ROLs to ROBs, etc.).

The general, implementation independent, ATLAS TDAQ architecture is presented. *Note that we have carried forward most of the design originally presented in the ATLAS TP (1994), DAQ/DCS/HLT TP (2000); note were choices have been made (e.g. level-2 requesting data)*

The architecture is presented in terms of the functional breakdown of the previous sections. Reference to chapter 2 is done to use / derive required performance figures (based on the design L1 rate of 75KHz, some reference to the expected behaviour at 100KHz as well?).

Detector read-out

ROL (Read-Out Link): the communication link out of the detector buffers (RODs). Each ROD may have one or more ROLs; each ROL corresponds to one event fragment. The ROL is expected to transport data at a rate equal to the maximum event fragment size times the maximum level-1 rate (i.e. XXX MB / sec).

RRC (ROD to ROB connection): the connection between the ROL and the ROB may be multiplexed, that is to say one or more ROLs may be connected to a single ROB. Hence a functional element in the system which represents how ROLs are multiplexed into ROBs. Figures on required bandwidths

ROB: the detector fragments are read out of the RODs and stored into TDAQ buffers; depending on the level of multiplexing provided by the RRC component, one or more fragments may be stored into a single ROB for the same event. *Figures on buffer depth; input and output bandwidth.*

RRM (ROB to ROS Multiplexor): in order to reduce the number of connections into the level-2 and event builder networks it is possible to funnel a number of ROBs into a single component. The RRM represents this ROB multiplexing capability. *Give figures on multiplexing capability (based on ROB output bandwidth).*

ROS (Read Out System): a component for serving data to the level-2 and event builder. It may also be used to introduce a further level of buffering before the event builder.

Level-2:

RoIB: the component which determines which fragments ought to be analysed by level-2 for a particular event, based on information received from the level-1 trigger. *(Note it runs at the L1 rate)*

L2SV: The level-2 trigger supervisor (L2SV): the component which, for a given event accepted by level-1, receives the information produced by the RoIB, assigns a L2PU to process the event and inputs the L2PU with the information provided by the RoIB.

L2PU: the component which, using the information produced by the RoIB, requests event fragments from the ROS, process them and produces a decision (accept / reject) for an event. The decision is passed to the ROS in order for this latter to remove (from the ROS buffers) or forward (to the final part of TDAQ, the event filter) the event.

L2N (level-2 network): the switching network used to connect all the ROSes, level-2 processors and supervisors for the purpose of moving ROI data and level-2 decisions between the TDAQ buffers, level-2 processors and supervisory components.Note that data and control share the same network. *Figures on expected bandwidths and rates.*

Event Builder

DFM: the supervisory element which assigns an event, accepted by level-2, to an SFI.

EBN (Event Builder Network): the event builder will handle events at a rate of a few KHz; to achieve this performance several events are built concurrently into many SFI's by means of a switching network which connects ROSes, SFIs and DFM. Note that data and (event builder) control share the same network. *Figures on expected bandwidths and rates*

SFI: the buffer where a full event is built prior to being moved to the event filter for further selection. *Target performance ~ 70MB/sec.*

Event Filter:

EFP (Event Filter Processors): A farm of processors, to run the algorithms; including possibly a supervisory component to assign events, available in the SFIs, to event filter processors. *Figure on expected time/event.*

EFN: A communication system connecting SFIs, event filter processing unit and SFOs. *Note that the issue here will be one of connecting a lot of processing units more than actual volumes of data (ratio processing to communication).*

SFO: A set of memories, sub-farm output (SFO), to buffer the events accepted by the event filter prior to writing the events to permanent mass storage. *Again expect performance ~ 70 MB/sec*

DCS (Detector Control System): *at this level of architectural detail the detector control system is seen as an unstructured entity which interfaces with the rest of TDAQ via the online network.*

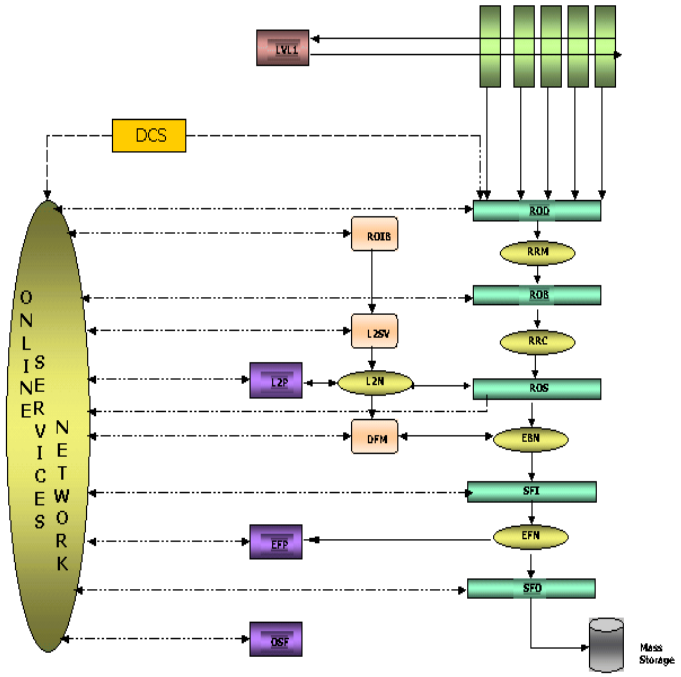
Online:

OSF (Online Software Farm): the farm of processors on which the TDAQ software services, such as the run control and the monitoring facilities. A single partitions and the whole experiment are operated out of this farm.

DBS (Data Base Servers): the set of servers used to hold the data bases.

OSN (Online Software Network): a network connecting the Online software farm, the detector control system as well as the controller and supervisors local to the TDAQ components. A more detailed organisation of this network, showing which TDAQ elements have a controller etc., is provided below in the detailed component views. *Some figures on expected performance and size of the network.*

The generic architecture is shown pictorially in Figures 5-2.



For missing details, one should refer to the "views" below to e.g. go into more details as regards their (the view's) parts. For example a ROB has a data flow view as buffer, but also a control view, a data base view.

The drawing has to be corrected to show LVL1 and its connection to the ROIB, the relationship L2SV/ DFM. Which one can do after we agreed upon the principle

5.5 TDAQ data flow architectural view

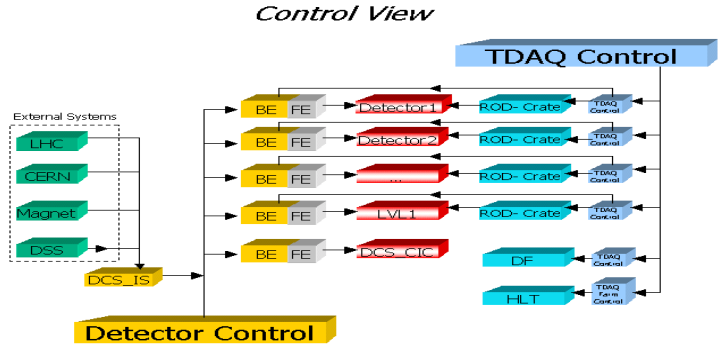
Specialise generic architecture for the purpose of Data flow.

Shall contain: functional decomposition into DF packages and sub-packages; interfaces and boundaries between DF packages and sub-packages; main use-cases realisation; "Event control and event flow" view which will include the rates and data volumes between DF packages and sub-packages (including type of communication).

5.6 TDAQ controls and supervision view

Specialised generic architecture for the purpose of control and supervision (eg show local controllers).

Includes both DCS and Online controls.



Remarks to this view:

This view has been chosen to illustrate the relation between the TDAQ Control and the DCS Control and explanations and some reasons for choosing this view are listed below.

- *Arrows represent the direction of command flow.*
- *Lines without arrows represent information exchange which is vital for correct decision making of the controlling master. Component boxes represent logical entities.*
- *Systems external to Atlas are shown where they have a vital importance to the experiment control. The controlling master must have knowledge of the machine status in order to take correct decisions.*
- *LHC: LHC machine status; CERN: Cern infrastructure; Magnet: Magnet status; DSS: Detector Safety System*
- *During data taking periods when TDAQ Control is active it has master control over the TDAQ system and the Detector control system.*
- *Outside data taking periods, when TDAQ Control is not active, the Detector Control system stays fully operational and controls all its connected units.*
- *Each detector can be controlled independently both from the TDAQ Control including the Detector Control during data taking periods, for example during installation and test phases, or outside data taking periods via Detector Control.*
- *The Command flow from TDAQ Control to Detector Control is performed from the TDAQ control on the level of detectors to the Detector Control on the same level.*

- *The presented components will be expanded and explained in more detail in the chapter on Experiment Control, when necessary details have been explained in the chapters on components and interfaces.*

5.7 Information sharing services view

Specialise the generic architecture for the purpose of information sharing services provided by the online software.

There are several arias where the information sharing is used in the TDAQ system: synchronisation between processes, error reporting, operational monitoring, physics event monitoring, etc. There are different types of information which TDAQ applications may share in different cases. The Online Software provides a number of services to support all the possible types of information exchange between TDAQ software applications.

As it is shown on Figure 5-2, each of those services acts as a common communication bus for all the TDAQ systems and detectors. Information can be shared between applications belonging to the same TDAQ system, among several TDAQ systems, and to each of the TDAQ systems and detectors.

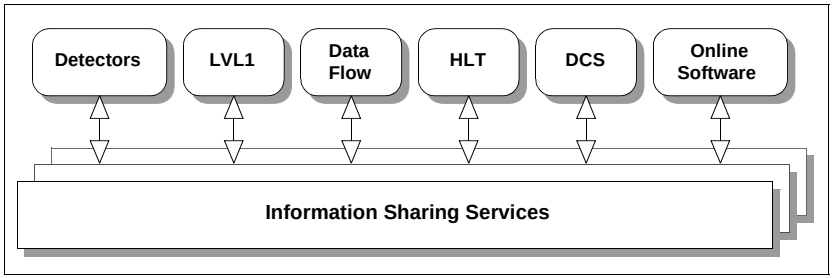


Figure 5-2 Information Sharing context diagram

All the Information Sharing services are partitionable in a sense that different instances of the same service are able to work in different TDAQ Partitions concurrently and fully independently.

5.8 TDAQ data base view

Data base architecture: including where access to (in and out of) databases is done.

Remark: This is a very basic view of the databases in TDAQ. It is expected that more details can be presented when a common understanding is reached on the sharing of non-event data across DCS, DAQ,

HLT and offline systems. This will be the topic of discussion at the next ATLAS week and the next Atlas Software workshop.

TDAQ and detectors are using configuration databases to describe their system topology and the parameters which are used for data-taking. A variety of configurations can describe and combine different combinations of existing partitions which are prepared for different types of runs (physics, calibration, debug, shutdown, etc.).

The TDAQ and detectors are using offline conditions databases to read and to store conditions under which the event data were taken. Such databases are used by the offline group for analysis and reconstruction of physics data and by the tdaq experts to analyse logs of the operational monitoring information stored during data taking by the online bookkeeper.

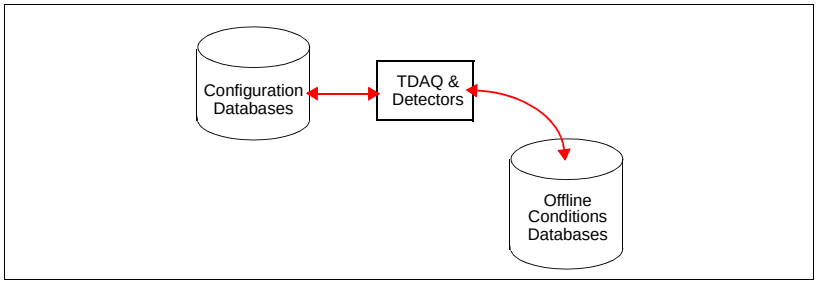


Figure 5-3

5.9 HLT view

The HLT issues relative to the generic architecture. It should probably include the organisation of the EF and LVL2 blobs, how HLT gets at the data.

5.10 Partitioning

The definition of partitions and the allowed operations are defined in Chapter 3, "System Operations". We remind that partitioning refers to the capability of providing the functionality of the complete TDAQ to a subset of the ATLAS detector.

The definition of the detector subset defines, because of the connectivity between RODs and ROBs, which ROBs belong to the partition. Downstream of the ROBs a partition is realised by assigning part of TDAQ resource (EBN, SFI, EF, online farm and network) to the partition: it is a resource management issue. In particular a subset of the ROBs, as mentioned above, and a subset of the SFIs *Note that this assumes that assigning SFIs implies associating a sub-set of the EF farm; if this is not the case then routing of events has to be done by the SFIs (see connection to TTC below).*

As regards the transport of the data across the allocated resources, the DFM plays the key role of routing subsets of ROBs to the associated subsets of the SFIs. In order for this to happen, in the case of partitions associated to non physics runs (i.e. when there is no level-2), the DFM must re-

ceive, via the TTC, the triggering information for the active partitions. *Need for connections of up to 35 TTC systems to DFM*

5.11 Baseline architecture implementation

The baseline architecture outlined in this section defines a concrete implementation for each of the components in the previous sections. The choices for the baseline are guided by the following criteria:

- Existence of working prototypes.
- Performance measurements which either fulfil the final ATLAS specifications today or can be safely extrapolated (e.g. CPU speed of commodity PCs).
- Clear evolution and staging path from an initial small system for use in testbeams and commissioning to the full ATLAS system.
- Overall cost-effectiveness and cost-effective implementation of a staging scenario.
- Possibility to take advantage of future technological changes over the lifetime of the ATLAS experiment.

The proposed baseline architecture is a system that can be build with today's technology and achieve the desired performance. It is expected that changes in the area of networking and computing will continue with the current pace over the next few years and will probably simplify various aspects of the proposed architecture. In addition optimizations, in particular in the area of the ROB I/O, will still be studied.

By making use of commercial off-the-shelf (COTS) components wherever possible the architecture will be able to take advantage of any future improvements in industry in a straightforward way. Only two custom components are foreseen in the final system: the RoIBuilder, of which only a single instance is needed, and the ROBins. The prototype for the latter is currently implemented on a single PCI board and about 800 to 1600 will be needed for the full system.

The performance numbers for the components which lead to a justification for the proposed architecture can be found in Part 3.

5.11.1 Overview

Some introductory text; note that largest part dedicated to dataflow; the figure also refers to dataflow only.

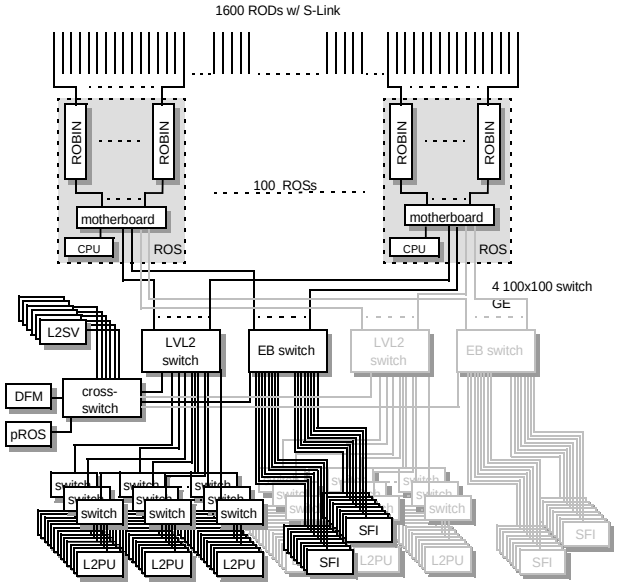


Figure 5-4 Baseline dataflow sketch

Insert a table summarising: components, their number, their expected performance.

5.11.2 Read Out Link

The ROD to ROB link will be SLink. Since the ROBs will be located near the RODs in USA15 these links can be rather short compared to other scenarios where ROBs live on the surface. About 1600 links will be needed.

Table 5-2

Component	Type (custom/ COTS)	Performance	Number of components	?	Comments
ROL					
ROB					
ROS					
L2N					
EBN					
L2PU					
RoIB					
L2SV					
SFI					
EFPU					
EFN					
SFO					
ONF					
OSN					
DCS					

5.11.3 Read Out Buffer

The Read Out Buffer (ROB) is implemented as a PCI board taking two input SLinks and a PCI output interface; it also includes a single GBit Ethernet output link. It is located in USA15 near the ROD crates.

The prototype ROB has X MByte of buffer memory. The high-speed input data path from the RODs is handled by an FPGA. An additional PowerPC CPU is available on each ROB.

A ROB buffers its input data until the Level 2 system has made its decision. During that time it will serve data to Level 2 on request.

The current prototype allows to access ROB data both via PCI and the Gigabit ethernet interface. The former (PCI) path is the baseline choice; the latter (Gigabit Ethernet) facility will be used to test alternative ways to optimize the flow of event data between the ROBs and the downstream networks.

The final ROB design is expected to support 4 ROL channels.The ROB realises (by a factor 4x1) the multiplexing indicated by the RRM function in the generic architecture.

5.11.4 Read Out System

The Read Out System (ROS) is a rack-mounted PC with multiple PCI buses and with multiple ROBs per ROS: today's scenario for the final system includes 3 ROBs per PC, each ROB with 4 ROLs. Requests for fragments coming for level-2 and requests for super-fragments (sequential merging of up to 12 fragments) for the event builder are handled by the ROS, i.e. by the PC.

Each ROS is provided with two Gigabit Ethernet interfaces, one towards the level-2 network, the second for the event builder network.

Optimization of the I.O, as indicated previously, will be further studied, for example in a scenario where I/O to level-2 is performed directly from the ROB via its Gigabit ethernet interface. This latter scenario will include also small switches concentrating several ROB channels into a small number of connections to the level-2 and event builder networks.

The ROSes are expected to be housed underground, so that the number of long fiber connections is of the order of twice the number of the ROSes.

5.11.5 Level 2 and Event Building Network

All data transferred between the ROSs and the Level 2 and event building units uses Gigabit Ethernet as the data link layer. The system includes a set of concentrating switches around the Level 2 units and a small set of central switches, for the level-2 and the event builder networks. The concentrating switches are used to reduce the number of ports on the central switches whenever the total bandwidth requirements of a component allows to bundle it with others without exceeding the capacity of a single Gigabit Ethernet fiber.

The baseline architecture assumes the use of Gigabit Ethernet interfaces throughout the system.

The level-2 and event builder central switches will, logically, be monolithic: in the sense that each switch will connect to all sources and destinations. However, they may be physically organised in terms of combined small switches.

Software-wise the event builder and level-2 networks are supported by the data collection component of the dataflow.

5.11.6 RoI Builder and Level 2 Supervisor

The RoIBuilder is a custom design described in ref[roibuilder]. It receives input from the Level 1 system via SLink. The output of the RoIBuilder is again sent via SLink to one of multiple Level 2 Supervisor processors. The latter are normal PCs connected to the data flow network.

The RoIBuilder design is modular and scalable: additional Level 2 Supervisor nodes can be easily added and will increase the performance linearly. Measurements done so far show that only a small number (< 5) of them will be needed.

The Level 2 supervisor is a PC running Linux. The only custom component is the SLink receiver card. Supervisor nodes can therefore be easily added and replaced in case of failure.

5.11.7 Level 2 Processing Units

Level 2 processing units will be normal rack-mounted PCs running Linux. Dual CPU systems are the most cost-effective solution at the moment, although that may change in the future. There will be about 130 Level 2 nodes in the final system. Initial setups during commissioning can be much smaller. Level 2 processing units are COTS components and can be replaced and added at any time.

5.11.8 Data Flow Manager

The Data Flow Manager (DFM) will be physically implemented by rack-mounted PCs running Linux. Again dual CPU systems are the most cost-effective solution at the moment.

Although there will be only one Data Flow Manager per partition, the need for multiple partitions may require a set of DFM nodes at any given time, all capable of performing the same task. All DFMs except for the one running the real trigger partition will need a TTC receiver board to receive detector specific triggers. DFMs receive input from the LVL2 Supervisor and talk to the SFIs and the ROBs.

5.11.9 SFI

The SFI components need no special hardware except for a second Gigabit Ethernet interface that connects them to the Eventfilter network.

The SFIs are rack-mounted PCs running Linux. The actual event building requires a lot of CPU capacity to handle the I/O load and the event assembly. Again dual CPU systems are the most cost-effective solution at the moment.

5.11.10 Eventfilter Network

The current baseline for the Eventfilter network is actually a set of small networks, each connecting a set of Eventfilter nodes with a small set of SFIs (possibly even one) and SFOs. This allows maximum flexibility in choosing the number of Eventfilter nodes for each cluster. The input rate can be increased by simply adding more eventbuilding node to a given subnetwork.

An optimization of the baseline option, above, includes a common event filter network connecting all SFI's, SFO's and event filter nodes together.

Since Eventfilter code will be CPU bound they will not require the full Gigabit bandwidth to keep them busy with data. Hence concentrating switches can be used to cluster multiple nodes and connect to a central switch.

5.11.11 Eventfilter Nodes

Eventfilter nodes are rack-mounted PCs, again most likely dual CPU machines. Computing performance is more important than I/O capacity for these nodes. Due to the large number of

CPUs required the use of blade servers which house hundreds of processors in a rack together with local switches might be a more attractive solution than 1U rack-mounted PCs.

5.11.12 Sub Farm Output

The Sub Farm Output (SFO) nodes take the accepted events from the Eventfilter nodes and pass them to mass storage. In the current implementation these are normal PCs with an attached hard disk. They write events to the disk in a series of files. It is assumed that a different process picks up those files and sends them to their final destination in the CERN computing division.

The SFOs also provide the necessary buffering if the network connection to the CERN main site is down. Assuming that the SFOs have to buffer up to 24 hours of event data at 200 Hz times 1.8 MByte, they will need a total of 32 TByte of disk storage. Today PC servers can be bought with > 3 TByte of cheap IDE disk storage for less than $10000. The SFOs will therefore be consist of normal PCs but with a housing that allows to add large disk arrays.

5.11.13 Other baseline elements

ONLINE software: controls etc.

DCS

5.12 Scalability of the system

Define required performance profile (in terms of level-1 rate vs time)

Discuss the scalability of the system, as a function of the Level-1 rate. First of all the main implication of a change in level-1 rate is in the HLT area: the size of the farms. This issue is both a cost driver and has implication on the size of the central networks. At low values of the level-1 rate,

In the baseline scenario, the detector read-out is present in its entirety since the beginning.

Scalability of the network may be achieved by using a single switch at low rates (sharing l2 and eb traffic).

Show diagrammatically the evolution of the system size as a function of level-1 rate.

5.13 References

5-1 Document from Architecture working group on global architecture.

5-2 DataFlow Architecture document.

5-3 ROS Architecture document.

5-4 Data Collection Architecture document.

6 Fault Tolerance and Error Handling

6.1 Fault Tolerance and Error Handling Strategy

Error handling and fault tolerance are concerned with the behaviour of the TDAQ system in the case of failures of its components. By failure we mean the inability of a component to perform its intended function. This includes both hardware and software caused problems.

The overall goal is to *maximize system up-time, data taking efficiency and data quality* for the ATLAS detector. This is achieved by designing a *robust* system that will keep functioning even when various parts of it are not working properly.

Complete fault tolerance is a desired system property which does not imply that each component must be able to tolerate every conceivable kind of error. The best way for the system to achieve its overall goal may well be to simply reset or reboot a component which is in an error state. The optimal strategy depends on the impact the faulty component has on data taking, the frequency of the error and the amount of effort necessary to make the component more fault tolerant.

The fault tolerance and error handling strategy is based on a number of basic principles:

- Minimize the number of single points of failure in the design itself. Where unavoidable, provide redundancy to quickly replace failing components. This might consist of spare parts of custom hardware or simply making sure that critical software processes can run on off-the-shelf hardware which can be easily replaced.
- Failing components must affect as little as possible the functioning of other components.
- Failures should be handled in a hierarchical way where first local measures are taken to correct it. Local recovery mechanisms will not make important decisions, e.g. to stop the run, but pass the information on to higher levels.
- All errors are reported in a standardized way to make it easy to automate detection and handling of well-defined error situations (e.g. with an expert system).
- All errors will be automatically logged and be available for post-mortem analysis if necessary. Where the error affects data quality the necessary information will be stored in the condition database.

We distinguish the following cases:

Error detection describes how a component finds out about failures either in itself or neighbouring components. Errors are classified in a standardized way and may be transient or permanent. A component should be able to recover from transient errors by itself once the cause for the error disappears.

Error response describes the immediate action taken by the component once it detects an error. This action will typically allow the component to keep working but maybe with reduced functionality. Applications which can sensibly correct errors that are generated internally or occur in hardware or software components they are responsible for should corrected them directly.

In many cases the component itself will not be able to take the necessary action about failures in a neighbouring component. Even if the component is unable to continue working, this should not be a fatal error for the TDAQ system if it is not a single point of failure.

Error reporting describes how the failure condition is reported to a higher level which might be able to fix the error condition. The mechanism will be a standardized service which all components use. The receiver of the error message might be persons (like a shifter or an expert) or an automated expert system.

Error recovery describes the process of bringing the faulty component back into a functioning state. This might involve manual intervention by the shifter or an expert or an automated response initiated by the expert system. The time-scale of this phase will typically be longer than the previous ones and can range from seconds to days (e.g. in the case of replacing a piece of hardware which requires access to controlled areas).

Error prevention describes the measures to be taken which prevent the errors to be introduced to hardware or software. Good software engineering, the use of standards, training, testing and the availability and use of diagnostic tools help in making the TDAQ system fault tolerant.

6.2 Error Definition and Identification

In order to respond to error conditions it is important to have a clearly defined TDAQ wide classification scheme that allows proper identification. It is assumed that error conditions are detected by data flow applications, controllers, event selection software and monitoring tasks. These conditions may be caused by failures of hardware they control, of components that they communicate with or these may occur internally.

The sources have a dual responsibility: correct anomalous conditions immediately or issue an error message, suitably classified and containing all necessary information for subsequent action by human or expert system.

Error messages are classified according to severity. The classification is necessarily based on local judgement; it is left to human/artificial intelligence to take further action, guided by the classification and additional information provided by the applications that detect the errors:

Additional information consists of a unique TDAQ wide identifier (note that status and return codes, if used, are internal to the applications), determination of the source and additional information needed to repair the problem. All messages are directed to an Error Reporting Service, never directly to the application that may be at the origin of the fault.

For successful fault tolerance, it is essential that correct issuing of error messages is enforced in all TDAQ applications.

6.3 Error Reporting Mechanism

Applications encountering a fault make use of an error reporting facility to inject an appropriate message to the TDAQ system. The facility is responsible for the message transport, message filtering and message distribution. Optional and mandatory attributes can be passed with the message. The facility allows receiving applications to subscribe to a message according to the

severity or other qualifiers independent of its origin. A set of commonly used qualifiers will be recommended. These can for example include the detector name, the failure type like hardware, network, software failures, or finer granularity indicators like ‘Gas’, ‘HV’ etc. They provide together with mandatory qualifiers like process name and id, injection date and time, and processor identification provide a powerful and flexible system logic for the filtering and distribution of messages.

Automatic message suppression at the sender level is foreseen to help avoiding avalanches of messages in case of major system failures. A count on the suppressed messages will be kept. Message suppression at the message reporting system level will also be possible.

An error message database may be used to help for standardization of messages including their qualifiers. A help facility could be attached which allow the operator to get detailed advise for the action on a given failure.

6.4 Error Diagnostic and Verification

Regular verification of the system status and its correct functioning will be a vital operation to help avoiding failures to occur. A customizable diagnostic and verification framework will allow to verify the correct status of the TDAQ system before starting a run or between runs, automatically or on request. It will make use of a suite of custom test programs which are specific for each component type in order to diagnose eventual faults.

6.5 Error Recovery

Error recovery mechanisms describe the actions which are undertaken to correct any important errors that a component has encountered and can not handle on its own. The main goal is to keep the system in a running state and minimize the consequences for data taking.

There will be a wide range of error recovery mechanisms, depending on the subsystem and the exact nature of the failure. The overall principle is that the recovery for a failure should be handled as close as possible to the actual component where it occurred. This allows both to isolate failures to subsystems without necessarily involving any action from other systems, to decentralize the knowledge required about properly reacting to a failure and to allow experts to modify the error handling in their specific subsystem without having to worry about the consequences for the full system.

If a failure cannot be handled by a subsystem at a given level, it will be passed on to a higher level in a standardized way. While the higher level will not have the detailed knowledge to correct the error, it will be able to take a different kind of action which is not appropriate at a lower level. For example it might be able to pause the run and draw the attention of the shifter to the problem, or it might take a subfarm out of the running system and proceed without it.

The actual reaction to the failure will strongly depend on the type of error. The same error condition, for example timeouts on requests, may lead to quite different actions depending on the type of component. A list of possible reaction is given in Chapter 6.7, "Typical Use Cases".

Each level in the hierarchy will have different means to correct failures. Only the highest levels will be able to pause data taking or decide when to stop a run.

The functionality for expert system type automatic recovery will be integrated into the hierarchical structure of the TDAQ control framework and can optionally take automatic action for the recovery of a fault. It provides multi-level decentralized error handling and allows actions on failures on a low level. A knowledge base containing the appropriate actions to be taken will be established at system installation time. Experience from integration tests and in test beam operation will initially provide the basis. Each supervision node can contain rules customized to its specific role in the system.

6.6 Error Logging and Error Browsing

Every reported failure will be logged in a local or central place for later retrieval and analysis. The time of occurrence and details on the origin will also be stored to help determining the cause and to build failure statistics, which should lead to the implementation of correcting actions and improvements of the system. Corresponding browsing tools will be provided.

6.7 Typical Use Cases

This paragraph will describe how a component would react to some typical faults both in a global approach and discussing any system specifics. The current collection is incomplete. Contributions should be provided by the systems. It is the intension to include details which would go beyond the scope (or space) in the TDR in a supporting document.

ROL (flow control, missing ROD fragments, failure); DF applications (failure of one or more); control and/or event data messages (packet loss, flow control, QOS (peer to peer or switches). Results from modelling may be used to justify.

A short list of possible reactions on different levels (from inside an application to system wide) and their impact on data taking follows:

- Symptom: Read-out link not working properly.
  - Action: Reset of local hardware.
  - Impact: Some parts of the event might be missing. If successful only an informational message would be send to the higher level. If not successful a error message would be issued.
- Symptom: Timeout for requests to a ROS inside a LVL2 node.
  - Action: Retry a configurable number of times.
  - Impact: Parts of an event might be missing. If not successful, the LVL2 trigger might not be able to run its intended algorithms and the event has to be force-accepted. If the error persists, the data taking efficiency might drop because the event building will be mostly busy with forced-accept events.
- Symptom: Dataflow component reports that ROS times out repeatedly.
  - Action: Pause the run, remotely reset the ROS component and if successful resume the run. If not successful, inform all concerned components that this ROS is no

longer available and inform higher level (who might decide to stop the run and take other measures like calling an expert).

- Impact: Data missing in every event.

- Symptom: LVL2 supervisor event request to LVL2 node times out.
  - Action: retry a configuration number of times. Then take node out of scheduler and report to higher level.
  - Impact: Available LVL2 rate is reduced
- Symptom: LVL2 Supervisor reports that LVL2 node repeatedly timed out.
  - Action: Remotely reset the offending node. If successful, the node should come back into the run. If not successful the
  - Impact: LVL2 rate is reduced while node is reset.
- Symptom: None of the nodes in a Eventfilter subfarm can be reached via the network (e.g. in case of a switch failure).
  - Action: Take all affected nodes out of any scheduling decisions (e.g. in the DFM) to prevent further timeouts. Inform higher level about the situation.
  - Impact: Data taking rate is reduced.

As can be seen, the same error condition (e.g. timeouts for requests) leads to quite different actions depending on the type of component. Each ROS is unique in that its failure leads to some non-recoverable data loss. A LVL2 node on the other side can be easily replaced with a different node of the same kind.

6.7.1 Reliability and fault tolerance in the Data Flow

This section was moved from Chapter 8, "Data-flow" and inserted as is here. It is most likely that this will be reworked into Section 6.7, "Typical Use Cases".

This section presents the major error use cases of the DataFlow. As a guideline to identifying the error use cases, major should be interpreted as referring to those that have directly influenced the design of the DataFlow. Each use-case is described as having transient, accumulative or persistent effect on the behaviour of the DataFlow. The handling of each use-case is presented based on results of real life tests. The exact layout of this chapter is subject to the identification of the major error use cases.

Each subsection groups related error use cases.

6.7.1.1 Detector read-out

Possible error use cases here are: ROL failure; assertion of one or more of the error bits in the S_LINK end of frame control word, i.e. ROD fragment corruption; assertion of S-LINK LDOWN; missing or out of sequence ROD fragments.

6.7.1.2 Level 1 to RoI builder

This sub section should be a summary of what is detailed in [8-28].

6.7.1.3 Control and event data messages

How the system handles the loss of each type of control message and event fragments separately.

6.7.1.4 Applications

How the system handles the failure of one or more of the applications.

6.7.2 Reliability and fault tolerance in the XXX system

describe important reliability and fault tolerant aspects in the respective system

6.8 References

Working Group Report of the Atlas TDAQ Error Handling and Fault Tolerance Working Group

7 Monitoring

7.1 Overview

A fast and efficient monitoring is essential during the data taking periods. Any malfunctioning part of the experiment must be identified and signalled as soon as possible so that it can be cured. The sources of monitoring information may be events, fragments of events or any other kind of information (histograms, counters, status flags, etc...). They may come from the hardware, processors or network elements, either directly or via the DCS. Some malfunctions can be detected by the sole observation of a single piece of information and could be performed at the level of the source of the information. An infrastructure has to be provided to process the monitoring information and bring the result to the end user (normally the shift crew).

The monitoring can be done at different places of the Data Flow in the DAQ system: ROD Crate, ROS, and SFI. Moreover, additional monitoring can be provided by the LVL2 trigger and by the Event Filter due to the fact that these programs will decode data, compute tracks and clusters, count relevant quantities for simple event statistics or to monitor the functioning of the various trigger levels and their selection power. It is possible to foresee that a part of the Event Filter is dedicated to monitoring activities where events tagged at read-out level or at previous stages of the selection are routed by the Data Collection (Event Builder) for special treatment. More generally, although this not relevant of the present chapter, one should envisage the case of an Online Farm downstream the Event Filter and in charge of performing CPU intensive monitoring tasks on events selected at the Event Filter level, e.g. calibration and alignment checks. This farm could also be the place where online calibration and alignment tasks are performed.

7.2 Monitoring sources

7.2.1 DAQ monitoring

7.2.1.1 Front-end and ROD monitoring

sub-detector front end electronics specific monitoring

- data integrity monitoring
- operational monitoring (throughput and similar, scaler histograms)
- hardware

7.2.1.2 Data Collection monitoring

DAQ specific monitoring

- data integrity monitoring
- operational monitoring (throughput and similar, scaler histograms)

- • hardware

7.2.2 Trigger monitoring

7.2.2.1 Trigger decision

simulate the decision of the trigger stages to confirm the quality of the decision

7.2.2.1.1 LVL1 decision

The LVL1 decision (for LVL1 accepts) is naturally cross-checked when doing the LVL2 processing. In addition, a pre-scaled sample if min-bias LVL1 triggers needs to be passed to dedicated processing tasks (possibly in a dedicated partition of the Event Filter).

7.2.2.1.2 LVL2 decision

Similarly to LVL1, the LVL2 decision (for LVL2 accepts) is naturally cross-checked when doing the EF processing and pre-scaled samples of events rejected at LVL2 should be passed to EF. Detailed information on the processing in LVL2 is appended to the event (via pROS) for accepted and force-accepted events. This will be available at the EF for further analysis.

7.2.2.1.3 EF decision

Detailed information should be appended to the event, for a sub-set of accepted and rejected events for offline further analysis.

7.2.2.1.4 Classification monitoring

In terms of monitoring, classification is a very important output of both LVL2 and EF processing. It consists of a 128-bit bitmap which records which signatures in the trigger menu were passed. Histograms can be filled locally on the processors where the selection is performed. With an accept rate of 1 kHz for LVL2 and 200 Hz for EF, and assuming a sampling rate of 0.1 Hz, a 1 byte depth is sufficient for the histograms. For both LVL2 and EF farms, the rate for the transfer of the histograms is therefore 1.2 kbyte / s.

7.2.2.1.5 Physics monitoring

The most simple approach to monitor the quality of the physics which is sent to permanent storage will consist in measuring the rates for some physics channel. It may be performed easily in the EF. A part of the results of these monitoring operations could be appended to the event bytestream for offline analysis. Others could be sent to the operator via the standard Online Software media for an online analysis.

Histograms of the rates for every item of the trigger menu as a function of time should be recorded, with the relevant variables with which they must be correlated (e.g. the instantaneous

luminosity). Such histograms can give very quickly an evidence for malfunctioning, although their interpretation may be quite tricky.

Well-known physics channels could be monitored so that one could permanently compare the observed rates with the expected ones. The list of such channels should be established in collaboration with the physics groups.

Information coming from the execution of reconstruction algorithms may be of interest. One could monitor e.g. the number of tracks found in a given detector on a per event basis. There again, a comparison with reference histograms may be of great help to detect malfunctioning. Physics quantities should be monitored, e.g. mass-plots of W and Z, n-Jet rates, reconstructed vertex location, quality of muon-tracks, quality of calo clusters, quality of ID tracks. Input is required from offline reconstruction groups.

7.2.2.2 Operational monitoring

Everything related to the “system” aspects, e.g. transportation of the events or event fragments, usage of computing resources, etc...

7.2.2.2.1 LVL1 operational monitoring

The integrity and correct operation of the LVL1 trigger will be monitored at both the hardware level by processes running in trigger crate CPUs and also by monitoring tasks in the Event Filter.

The LVL1 trigger is the only place where every bunch crossing is processed and where a crude picture of the real beam conditions can be found. For example, the calorimeter trigger fills histograms, in hardware, of the "level 0" rates and spectra of every trigger tower and can quickly identify, and if necessary suppress, hot channels. Hardware monitoring is also used to check the integrity of links between the successive steps in the trigger processor pipeline.

At the Event Filter, monitoring tasks will check for errors in the trigger processors at a lower rate than hardware monitoring, but with greater diagnostic power. Event Filter tasks will also produce various histograms of trigger rates, their correlation and history.

7.2.2.2.2 LVL2 operational monitoring

The LVL2 selection software runs as part of the Data Collection (DC) in the L2PU [L2MonitMRS]. It will therefore use the DC infrastructure and hence the monitoring tools foreseen for this system. The following aspects, relevant of DC, will be monitored:

- trigger, data and error rates
- CPU activity
- queue occupancies (load balancing)

One could also mention:

- LVL2 selectivity per LVL1 trigger type
- RoI sizes

- RoI occupancies per sub-detector
- RoI specific hit-maps per sub-detector

Monitoring of the quality of the data by LVL2 processors is not envisaged. Indeed, the available time budget is limited because of the necessity to release data from the ROB. Monitoring a fraction of the events in the L2PU is not desirable since this would introduce large variations in LVL2 latencies as well as possible points of weakness in the LVL2 system. The necessary monitoring of the LVL2 quality shall therefore be delegated to the downstream monitoring facilities, i.e. the EF (or online monitoring farm) and the offline analysis. One should however discuss very carefully the opportunity to fill in L2PU some histograms, possibly read at the end of the run, so that a high statistics information is given, which could not be reasonably be obtained by using forced accepted events on a pre-sampled basis. The evaluation of the extra CPU load for such operations should be made.

7.2.2.2.3 EF operational monitoring

The monitoring of the data flow in the Event Filter will be primarily done directly at the level of the EFD process. Specific EFD tasks, part of the main data flow, will be in charge of producing relevant statistics in terms of throughput at the different levels of the data flow. They have no connection with other processes external to EFD.. The detailed list of information of interest for the end user has not yet be finalised and will continue to evolve all along the lifetime of the experiment.

Among the most obvious parameters which are going to be monitored, one might quote:

- the number of events entering the Farm
- the number of events entering each sub-farm
- the number of events entering each processing host
- the number of events entering each processing task
- the number of events selected by each processing task, as a function of the physics channels present in the trigger menu
- the same statistics as above at the level of the processing host, the sub-farm and the Farm

Other statistics may be of interest such as the size of the events, as a function of different parameters (the time, the luminosity of the beam, the physics channel). As stated above, the identification of these statistics will be formulated more precisely at a later stage of the development of the experiment.

The results of the data flow monitoring will be sent to the operator via standard Online SW media (e.g. IS or Histogram Service in the present implementation).

7.2.2.2.4 PESA SW operational monitoring

A first list of parameters which could be monitored for debugging purpose and comparison with modelling results can be given:

- time spent in each algorithm
- frequency at which each algorithm is called

- number of steps in the step sequencer before rejection
- info and debug messages issued by the PESA SW
- number of active input/output trigger elements

Some of these points could be also monitored all along the duration of normal data taking.

Hooks necessary for these profiling measurements should be implemented directly at the level of the base class of the GAUDI algorithm. Profiling tools such as NetLogger for coarse measurements and TAU have already been studied in the context of LVL2 and their use at a larger scale will be considered in PESA SW.

It is intended to make use of the ATHENA Histogramming service, which should therefore be interfaced to the EF infrastructure. Some control mechanisms should be provided to configure the various monitoring options and to operate on the histograms (e.g. to reset them after having been transferred).

7.2.3 Detector monitoring

The detector monitoring can be done at different places of the Data Flow in the DAQ system: ROD Crate, ROS, and SFI. Moreover, additional monitoring can be provided by the LVL2 trigger and by the Event Filter due to the fact that these programs will decode data, compute tracks and clusters, count relevant quantities for simple event statistics or to monitor the functioning of the various trigger levels and their selection power.

The ROD level is clearly the one where the first check for monitoring the data quality and integrity can be easily done. The computing power provided by e.g. DSPs installed directly on the ROD board allows to perform sophisticated calculations and to fill histograms. Transportation of these histograms towards analysis workstations would then be performed by the ROD crate CPU (using the Online SW tools running on this CPU).

An extended part of the detector is available at the ROS level, and monitoring at this level is therefore considered as a potentially interesting facility. A correlation between ROS crates is not seen as needed because such a correlation may be obtained at the SFI level. Event fragments sampled at the level of the ROS could then be routed to dedicated workstations operated by the shift crew.

Some detectors will need a systematic monitoring action at the beginning of the run to check the integrity of the system. This concept has been already introduced at the test beam by the Pixel sub-detector: at the beginning of the run and at the end there are special events with a start and end of run statistics. The need of having this first check at the ROD level is driven by the huge amount of information. If monitored later would complicate the back tracking of possible problems. The frequency of this activity can be limited at the start and end of run in normal operations.

When information from several detectors is needed, the natural place to make it is obviously after the Event Builder. The SFI is the first place were fully assembled events are available. The monitoring at the level of the SFI is then the place where calorimetry, muons and level 1 wants to have the first cross check of consistency between the LVL1 information and the raw values of the trigger towers. Moreover at the SFI level a first correlation among sub-detectors is possible and is seen as extremely useful to spot all problems that not necessarily need a full reconstruction. At that level is also possible to perform a first quick correlation between different sub-detectors data, as for example checking the rough matching of a muon detected by the Muon precision chambers and a track in the Inner detector.

When monitoring require to perform some reconstruction operations, it seems natural to try to save computing resources by re-using the results obtained for selection purposes and therefore to execute this activity in the framework of the EF. In addition, some detectors foresee to perform monitoring at the level of event decoding, i.e. in the bytestream conversion service, and to fill histograms during the reconstruction phase associated with the selection procedure in the EF. These histograms should be sent regularly to the shift operators and archived. More sophisticated monitoring operations might require longer execution times and be potentially dangerous for the EF as robustness of the implementation with respect to crashes due to errors is not necessarily as well established as it could be for selection algorithms. The infrastructure provided by the monitoring tasks implemented in the EF (see Chapter 9) is intended to solve this problem.

Finally, some monitoring activity such as calibration and alignment checks may require events with a special topology selected at the level of the Event Filter. For instance, the Inner Detector group envisages to perform online the alignment of the tracking system. This requires some thousands of selected events, either stored on a local disk of fed directly to the processing task. Then CPU intensive calculations are required to invert matrices which may be as large as 30000 x 30000. With a cluster consisting of 16 PC (as available in 2007, i.e. 5GHz CPU clock, 1GB of fast memory and 64 bit floating point arithmetic unit), this can be made in less than one hour. A very efficient monitoring of the tracker alignment can therefore be performed.

7.3 Monitoring destinations and means

This section aims at describing where and how (i.e. with which tools) it is intended to perform monitoring operations

7.3.1 Online Software services

The Online Software (see Chapter 10) provides a number of services which can be used as monitoring mechanism which is independent of the main data flow stream. The main responsibility of these services is to transport the monitoring data requests from the monitoring destinations to the monitoring sources and to transport the monitoring data back from the sources to the destinations.

There are four services provided for a different types of the monitoring information:

- **Event Monitoring Service** - is responsible for transportation of physical events or event fragments sampled from well-defined points in the data flow chain to the software applications which can analyse them in order to monitor the state of the data acquisition and the quality of physics data of the experiment.
- **Information Service** - is responsible for exchange of user-defined information between TDAQ applications and aimed to be used for the operational monitoring. It can be used to monitor the status and various statistics data of the TDAQ sub-systems and their hardware software elements;

- **Histogramming Service** - is a specialisation of the Information Service with the aim of transporting histograms. It accept several commonly used histogram formats (lik3 ROOT histograms for example) as the type of information which can be send from the monitoring sources to the destinations;
- **Error Reporting Service** - provides transportation of the error messages from the software applications which detect these errors to the applications which are responsible for their monitoring and handling.

Each service offers the most appropriate and efficient functionality for a given monitoring data type and provides specific interfaces for both monitoring sources and destinations.

7.3.2 Monitoring in the Event Filter

From the beginning of the design of the EF, it has been foreseen to perform some monitoring activities in it, in addition to the ones related directly to the operation. EF is indeed the first place in the data taking chain where the full information about the events is available. Decisions from the previous levels of the trigger system can be checked from both accepted and rejected (on a pre-scaled basis) events. Information coming from the reconstruction phase, which generally requires a large amount of CPU power, can be rather easily re-used, leading to large savings in terms of computing resources. Finally, the fact that EF is part of the data taking chain ensures that pertinent information will be made available to the shift crew in the best time delays.

Monitoring in the Event Filter, or more generally monitoring after the Event Builder, can be performed in different places:

- directly in the filtering tasks (which raises the problem of the robustness of the monitoring code),
- in dedicated monitoring tasks running in the context of the Event Filter (then, one should think of passing the information gathered in the filtering task to take profit of the already used CPU)
- or in a dedicated, independent sub-farm. In that case, that is the Data Collection DFM which takes care of directing potentially interesting events (tagged at read-out, LVL1 or LVL2 levels).

Those different possibilities are not mutually exclusive.

7.3.3 Monitoring after the Event Filter

In order to perform the CPU intensive monitoring activities described at the end of Section 7.2.3, some dedicated online farm must be provided. It would be fed by events specially selected in the Event Filter. Dedicated output channels may be devoted to such events which would be directed towards the farm. This farm could be also the place where online calibration and alignment tasks are performed.

7.4 Archiving monitoring data

Data which is produced by monitoring activities should be archived by some bookkeeping service so that it can be cross-checked offline with more detailed analysis. One should also store (in a dedicated channel?) events whose acceptation has been forced at any level of the selection chain. These events are necessary to evaluate precisely the acceptance of the trigger.

Part 2

System Components

8 Data-flow

8.1 (Possible introduction)

8.2 Detector read-out and event fragment buffering

8.2.1 Read-out link

Each sub-detector reads data from the detector over Front-End links and uses a ROD to multiplex the data. Each of the sub-detectors has different requirements (and different designers;-) consequently the implementation of the ROD varies between sub-detectors. The guidelines for designing the ROD are set out in the Trigger & DAQ Interfaces with Front-End Systems: Requirement Document [8-1]. The purpose of the ROL is to connect the sub-detectors to the TDAQ system and it is responsible for transmitting error-free data from the output of the ROD to the input of the ROS, the first element in the TDAQ chain.

The ROL requirements have been stable since the High-level Triggers, DAQ and DCS Technical Proposal TP [8-2]:

- 32 bit data at 40.08Mhz, (~160 MByte/s)
- A control bit to identify the start and end of an event
- Xon/Xoff flow control
- Error detection, error rate < 10^{-12}
- A maximum length of 300m for the fibre version, 25m for the electrical version.

To ensure homogeny, the output of the ROD is defined by the S-LINK specification [8-3]. In addition, The raw event format [8-4] defines the order and content of the information transmitted from the ROD. At the other end of the ROL, the ROS inputs are identical for all sub-detectors and also conform to the S-LINK standard.

The S-LINK specification has been stable since 1996. It is used in COMPASS and in other LHC experiments, e.g. CMS. S-LINK is an interface definition; it only defines protocols and recommends connector pin-out. As shown in Figure 8-1, the ROD end of the ROL is called the Link Source Card (LSC) and the ROS end the Link Destination Card (LDC). They are connected by optical fibres or copper cables. Event data flows from the LSC to the LDC on the forward channel. Flow control information, i.e. the ROS can stop the ROD sending data if it's input buffers are almost full, flows from the LDC to the LSC on the return channel.

The DIG - ROD Working Group have also recommended that the LSC be placed on a mezzanine card to facilitate support and upgradeability [8-5]. The form factor of these mezzanine cards is based on the CMC [8-6] standard.

The LSC plugs onto the S-LINK connector on the ROD (or its associated rear transition card). For the forward channel, a Field-programmable gate array (FPGA) handles the protocol and delivers words to a serial/deserialiser (SERDES) chip which performs parallel-to-serial data con-

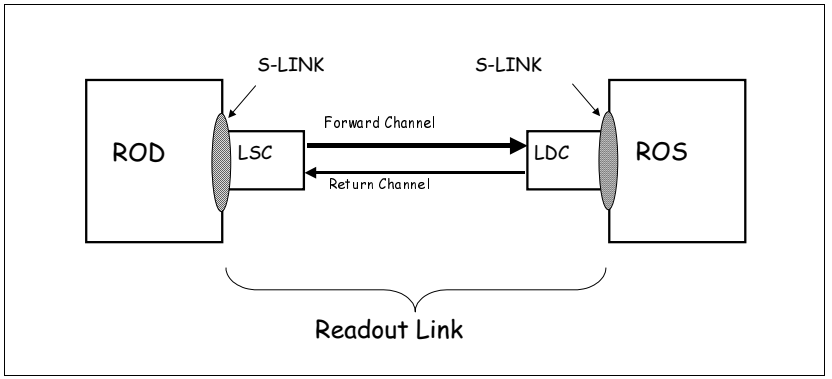


Figure 8-1 The relationship between the S-LINK and the ROL.

version and encoding. The output of the SERDES drives an optical transceiver that in turn feeds the optical fibre. The operation of the receiving card, the LDC, is a mirror image of the LSC. In fact the current LSC and LDC are physically the same card with different programs in the FPGA.

Various prototype implementations of the ROL have been built to prove the concept and measure the performance. The previous version of the ROL, the ODIN, used a physical layer that was based on the Hewlett Packard G-LINKs (HDMP-1032/1034). They have also been used successfully in ATLAS test-beams and eighty of these ROLs are being used in the COMPASS experiment. However, the maximum bandwidth is limited by the G-LINK at 128 MByte/s. Following the second ROD workshop, the requirements of the ROL were increased to 160MByte/s and a second version of this link was designed which used two G-LINKs chips per channel. Unfortunately, this raised the cost as two pairs of fibres and associated connectors were required.

Another recommendation of the ROD Working Group was to build a ROL that would use only one pair of fibres. This has been achieved by using 2.5 Gbit/s components in the current design, the High-speed Optical Link for ATLAS (HOLA) [8-7]. In this implementation a small FPGA, the EP20K30E APEX 20K, handles the S-LINK protocol. The SERDES chip is a Texas Instruments TLK2501 running at 2.5 Gbit/s for both for the forward and for the return channel (one per card). For the optical transceiver, the Small Form Factor Pluggable (SFP) Multimode 850 nm 2.5 Gbit/s with LC Connectors is recommended, e.g. the Infineon V23818-N305-B57. The use of pluggable components allows the optical components to be changed in case of failure.

Test equipment has been developed for the ROD/ROL/ROS. This includes an emulator that can be placed on the ROD to check that the ROD conforms to the S-LINK specification. Similarly, an emulator exists that can be placed on a ROS to emulate a ROL connection. The emulators allow ROD, ROL and ROS designs to be tested at full bandwidth and errors to be introduced in a controlled manner. The HOLA was produced and tested in 2002 and satisfies all requirements of the ROL.

In addition, for the purposes of exploitation in laboratory test set-ups and in test-beams, i.e. further testing, cards exist which allow the ROL to be interfaced to the PCI Bus in a PC. Performance measurements of this interface [8-8] have shown that data can be transferred into a PC at 160 MByte/s using a single ROL input. Modifications to the firmware have allowed the emula-

tion of an interface with four ROL inputs. Measurements using this emulator have demonstrated a bandwidth of 450 MByte/s into a PC. The next version of the interface, the FILAR, will have four ROLs on-board and should be ready for the April 2003 test-beam.

The purchase of the cards, in small quantities, is handled by the CERN stores. For quantities required for ATLAS a tendering process will be initiated in 2003 thus ensuring the availability of larger quantities during 2004. The production schedule will be adapted to the requirements of the sub-detectors who have been asked by the DIG to provide estimates of quantities for the years up to the start of the LHC. Maintenance and short-term loans of equipment will probably be handled by EP/ESS.

8.2.2 Read-out subsystem

8.2.2.1 High Level Design

The ROS has three major components: the RobIn, the IOManager and the LocalController. Figure 8-2 shows the relationship between the three ROS components and any other relevant TDAQ component. A complete high level design of the ROS can be found in [8-10], only a summary is presented here.

The RobIn component provides the temporary buffering of the individual data fragments produced by the RODs. All incoming ROD event fragments are buffered for the duration of the LVL2 trigger decision time. It thus needs to receive and buffer incoming ROD event fragments at the full LVL1 trigger rate. The ROD event fragments are buffered The date In addition, also sends on request the accepted ROD event fragments.

Due to these very demanding requirements the baseline RobIn is designed and implemented as a custom hardware module. Section 8.2.2.2 describes the high level design of the RobIn component and the results of measurements obtained with a prototype RobIn.

The main function of the IOManager is the servicing of requests for data by the High Level Triggers. According to the criteria specified in the data request, the IOManager collects ROB fragments from one or more RobIns and builds a ROS Fragment. This fragment is then sent to a destination specified in the original request. It also receives from the High Level Triggers the request to release buffer space occupied by ROD event fragments. These event fragments have either been rejected by the LVL2 trigger or successfully built by the Event Building. So as to maximise the overall performance of the ROS, the design of the IOManager allows a number of data requests and releases to be handled concurrently that is to say, it overlaps I/O operations with the processing of requests.

In the baseline TDAQ implementation the IOManager units are implemented as multithreaded software processes.

The LocalController also provides a single interface point between the ROS and the Online software (configuration database, run control, message reporting, monitoring and process control). In particular the LocalController is responsible for retrieving the relevant information from the configuration database and sending it to the IOManager component. The IOManager also provides for the configuration, control and operational monitoring of its associated RobIns.

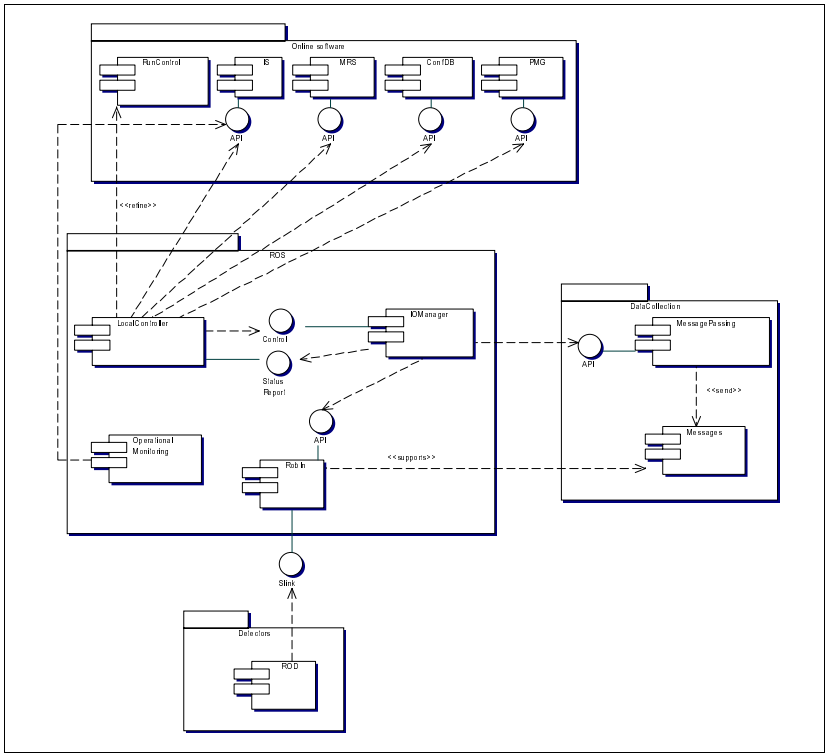


Figure 8-2 Main ROS components and their relationship with the other TDAQ components.

As the IOManager and the RobIn components, the LocalController are organized in specific units, each of which is connected to one or more IOManager units. In the TDAQ baseline implementation the LocalController is implemented as a multithreaded software processes.

In one of the possible future ROS deployment scenarios there would not be any IOManager component and all the RobIn units would be directly visible outside of the ROS system. In such a scenario all the control, monitoring, error handling functionalities provided by the IOManager will have to be provided by the other components or sub-systems, i.e. LocalController and DataCollection. No data multiplexing will be provided at the level of the ROS, and the different systems will always have to send data requests to the individual RobIn units and handle the individual ROB Fragments coming back from all of them.

8.2.2.2 Design of the ROBIN

The RobIn is located at the boundary between the detectors and the ROS. It receives ROD event fragments from a number of ROLs. Its basic functionality can be described by the following four tasks:

- Receive ROD event fragments from the ROL
- Buffer ROD event fragments
- Send ROD event fragments, on request, to the High Level Triggers
- Release ROD event fragments, on request, from the buffer.

Figure 8-3 shows the context of the RobIn.

Figure 8-3 Context of the RobIn.

The baseline RobIn takes into account the experience and results of studies from previous prototyping studies [8-12], [8-13] and the requirements on it are documented in the ROS-URD [8-9]. The final design of the RobIn will based on the final prototype whose complete design is documented in a set of documents [8-14], [8-15] and [8-16]. Here only a summary description is given.

Andreas: I intend to omit all the details of the function description which can be taken from HLDD and DLDD.

Editor: A reduced description needs to be given here.

Andreas: Should we elaborate on the issues of changing the ROL to GE, multiplexing etc?

Editor: What we finally put here should reflect the baseline choice, not an either or.

Referring to Figure 8-4, the primary functions of the RobIn (receive, buffer, send and release) are mapped onto a small number of specialised building blocks: ROL-IF - CORE - MEM - TDAQ-IF. It supports two ROLs, the data from which are buffered in separate buffers. All functionality related to the receiving of ROD fragments from the ROLs are realised in an FPGA, the CPU handles the issues related to management and monitoring.

The TDAQ-interface block implements two interfaces, a PCI bus and a network interface, allowing various DataFlow architecture options to be studied. The design of the final RobIn can be realised by removing (and not by adding) functionality, i.e. the PCI bus or network interface.

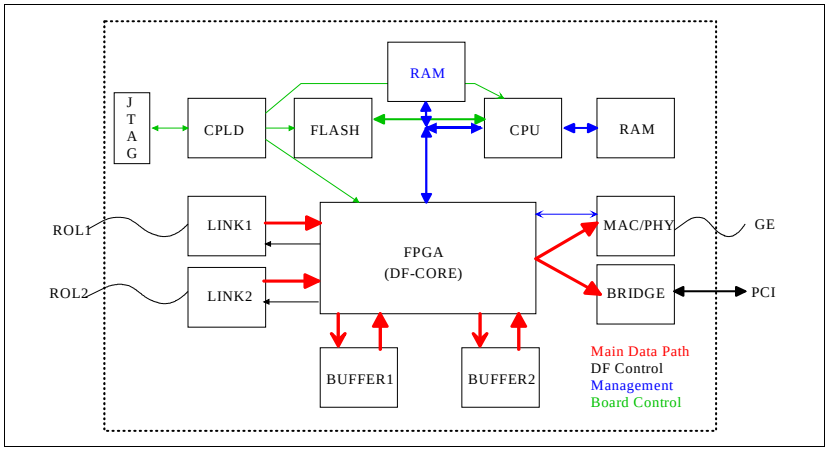


Figure 8-4 Basic Functional Diagram.

The Software Interface [8-16] to the prototype RobIn is generic enough to allow the RobIn to be studied in all DataFlow architectural options. The same basic services and messages are used in all cases, and functionality particularly required for e.g. the network is encapsulated in appropriate modules. The software interface comprises mainly the definition of a set of services provided by the RobIn and the messages to request these services and to transfer the responses.

The ROS LocalController, via the IOManager, performs the configuration and control of the RobIn via the TDAQ-interface block.

Andreas: Should we add more information about the production status?

Editor: It will be out of date at the time of print.

8.2.2.3 Implementation and performance

The baseline deployment of the ROS is shown in Figure 8-5.

Benedetto: for now I put the bus based ROS but this may change!!!!

It is deployed on two nodes: a ROS PC and a RobIn module. The former is a desktop PC running the Linux operating system and has at least one Ethernet connection for the purpose of communication with the Online system. In addition, it has at least four 64 bit / 33 MHz and 3.3 V PCI slots. These slots are used to host the RobIn modules. Each RobIn node has two SLINK input connections. The IOManager via the DC Message Passing interface receives data requests and release messages, and returns ROS event fragments to the High Level Trigger components. These communications occur via Ethernet.

Figure 8-6 shows an alternative way of deploying the same ROS components. In this case the RobIn devices are now implemented on dedicated boards, e.g. VMEbus 9U, and the connection with the ROS PCs is made through an Ethernet switch. The figure also shows that in this scenar-

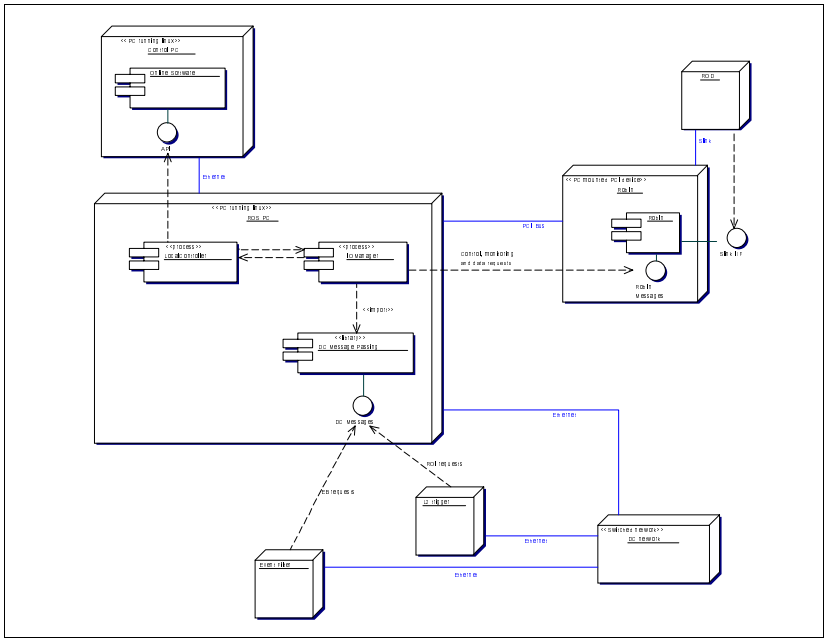


Figure 8-5 Baseline deployment of the ROS.

io the LVL2 trigger requests data fragments directly from the RobIn modules, without passing through the IOManager process.

Extensive measurement have been made on the performances of the ROS for the deployment scenarios previously described, bus-based and switch-based, here only the main results are presented, the complete set of results can be found in [8-20].

Figure 8-7 shows the setup for the bus-based testbed. In this testbed an IOManager and a LocalController process were deployed on a standard 2 GHz Xeon PC with a single processor and a 66 MHz / 64Bytes PCI bus, running RedHat Linux 7.3.

As the final prototype RobIns were unavailable at the time of these measurements, I/O with a RobIn was emulated using a number of RACE boards [8-19] which have the same physical PCI bus interface as the final prototype RobIn, and thus provide a very accurate emulation of the final devices. The RACE boards were not connected to any external data source and were programmed to generate ROB Fragments on demand.

The testbed has been run both in a standalone configuration, where the IOManager was generating triggers internally and the produced ROS Fragments where sent nowhere, and in a configuration where the IOManager was receiving real data request messages from the network and sending back the ROS Fragments to the requester process.

Figures 8-8 and Figures 8-9 show the maximum LVL1 rate that an IOManager was able to sustain for different fractions of LVL2 and Event Building requests and for different number of

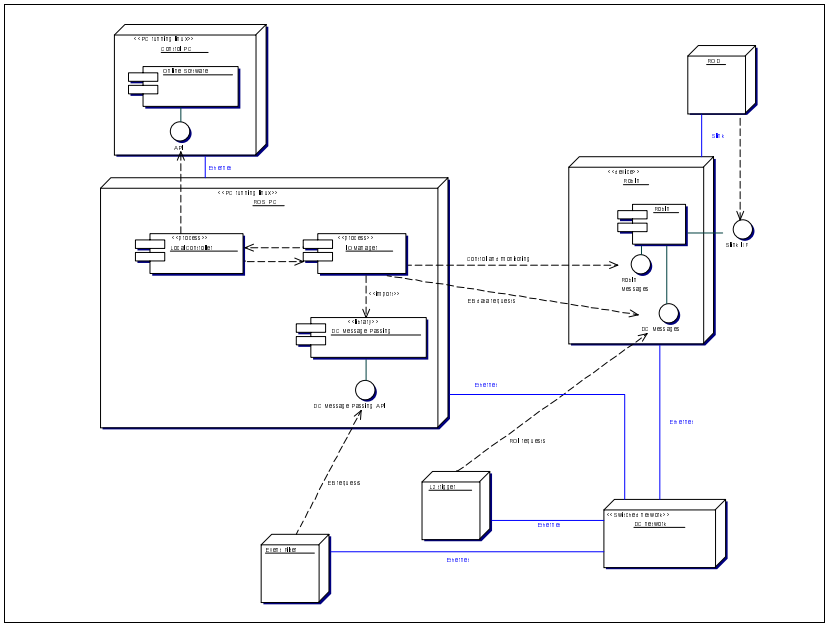


Figure 8-6 Alternative ROS deployment scenario.

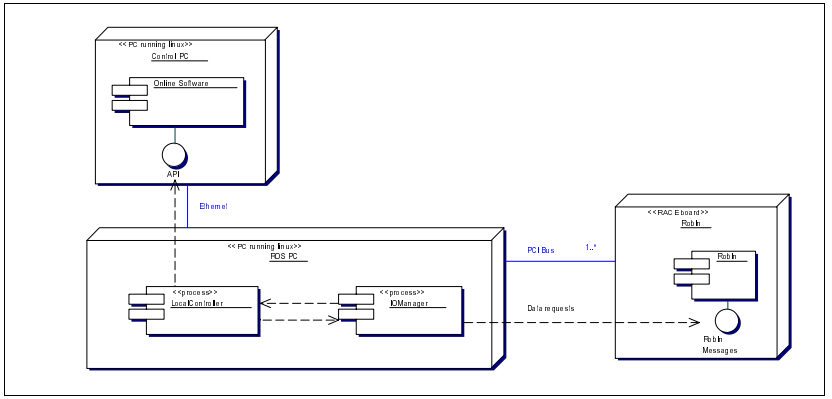


Figure 8-7 Setup of the testbed for studying the bus-based ROS system.

RACE boards connected to it. As we had only 6 RACE boards available, we developed a software simulation of the RobIn to allow the test of the IOManager performances with larger numbers of connected RobIn modules

The "software emulation" probably needs to be elaborated.

..

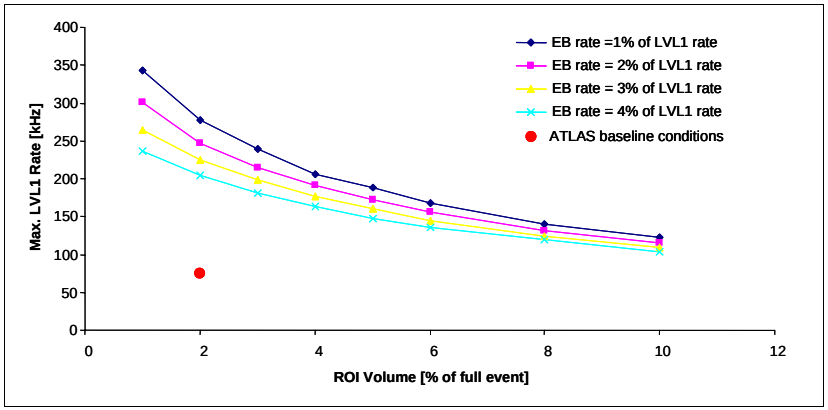


Figure 8-8 Maximum sustainable LVL1 rate for different fractions of LVL2 and Event Building requests, for a standalone bus-based ROS connected to a different number of RobIn modules.

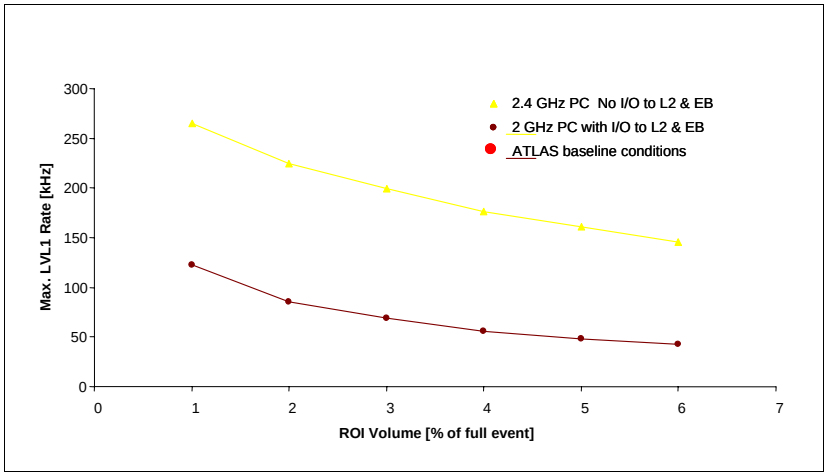


Figure 8-9 Maximum sustainable LVL1 rate for different fractions of LVL2 and Event Building rate of 3%, for a standalone bus-based ROS with real I/O to other DataFlow components.

Figures 8-10 shows how the simulation reproduces the measured results for up to 6 RobIn modules and the results that one obtains for a larger number of RobIns.

Figure 8-10 Maximum sustainable LVL1 rate for different fractions of LVL2 and Event Building requests for a standalone bus-based ROS with simulated PCI RobIn input.

The system is thus shown to fulfil the requirements for the final ATLAS for up to XXXX PCI RobIn modules connected to a same IOManager.

Figures 8-11 shows the setup for the switched-based testbed. The IOManager and a LocalController process were deployed on a standard PC and the RobIns were emulated with a number of FPGA emulators [8-21] that were programmed to receive data requests over the network with the same message passing interface as the final RobIn prototype and to generate ROB Fragments on demand. Similarly to the tests performed on the bus-based ROS, this testbed has been operated both in a standalone configuration, where the IOManager was generating triggers internally and the produced ROS Fragments where sent nowhere, and in a configuration where the IOManager was receiving real data request messages from the network and sending back the ROS Fragments to the requester process.

Figure 8-12 and Figure 8-13 show the maximum LVL1 rate that the system was able to sustain for different fractions of LVL2 and Event Building requests and different number of connected RobIn sources. Also in this configuration the system is shown to fulfil the requirements for the final ATLAS, see SOME SECTION IN PART 1..

8.2.2.4 pROS

Main description here. Short addition in Chapter 9, "High-level trigger".

8.2.3 ROD crate data acquisition

The ROD is a sub-detector specific front-end element. It is located, in the event data flow, after the first level of on-line event selection, between the Front-end Electronics (FE) and the ROS. The ROD receives data from one or more Front-end Links (FELs) and sends data over the ROL

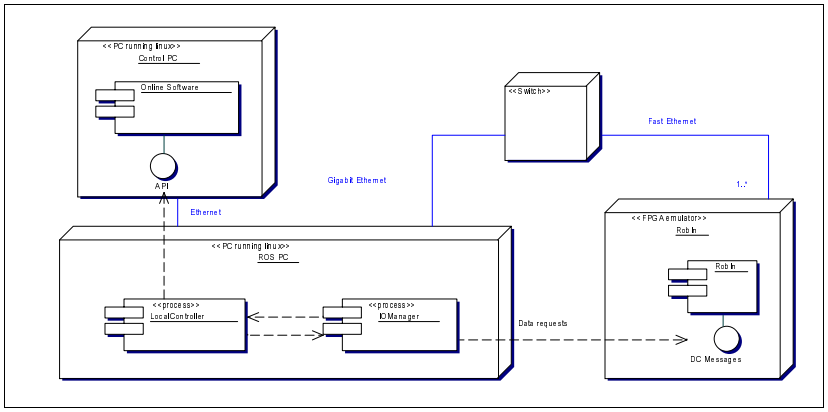


Figure 8-11 Setup of the testbed for studying the switched-based ROS system.

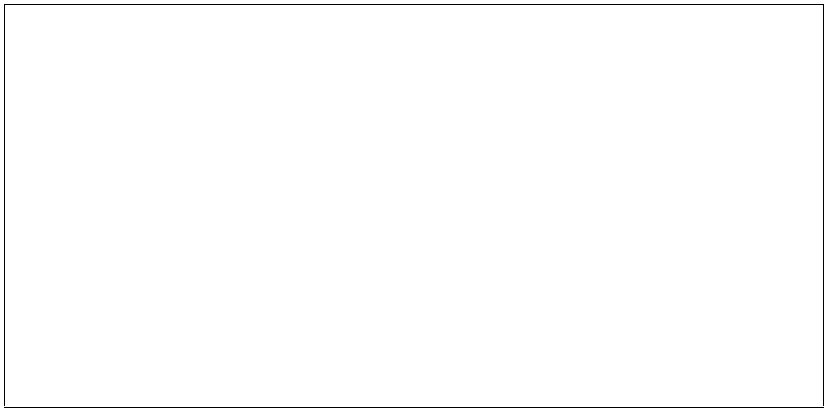


Figure 8-12 Maximum sustainable L1 rate for different fractions of L2 and EB requests, for a standalone network-based ROS connected to a different number of RobIn modules.

to the ROB. The ROD System covers all RODs and other functional elements at the same hierarchical level in the event data flow between the FE and the ROS. Those elements are grouped in crates. The crates contain ROD Crate Modules (RCMs) which can be: RODs, modules other than RODs, e.g. for control of the FE, for processing event data upstream of the RODs or for driving a TTC partition, as well as not fully functional ROD prototypes in laboratory setups or at test beam, and one or more ROD Crate Processors (RCPs). Each ROD Crate is connected to one or more ROD Crate Workstations (RCWs).

The sub-detectors need common DAQ functionality at the level of the ROD Crate for single or multiple ROD Crates in laboratory setups, at assembly of detectors, at test beam, and at the experiment during commissioning and production. ROD Crate DAQ [8-17] is part of the TDAQ system. It comprises all software to operate one or more ROD Crates and runs inside the ROD

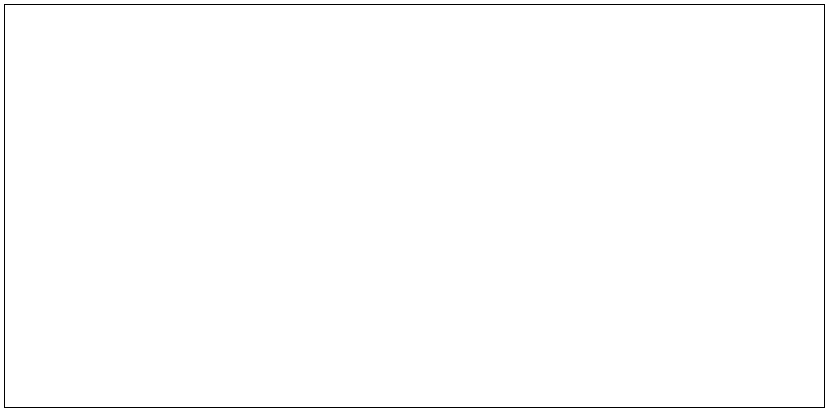


Figure 8-13 Maximum sustainable L1 rate for different fractions of L2 and EB requests, for a network-based ROS with real DC I/O.

Crate as well as on the RCWs. It provides the functionality for configuration and control, ROD emulation, monitoring, calibration at the level of the ROD Crate, and event building across multiple ROD Crates.

8.2.3.1 High Level design

The ROD Crate configuration describes all necessary data required to fully configure all modules of the ROD Crate and the RCW(s). All ROD Crate configuration data are stored in one or more databases with the configuration database of the Online Software being the driving one. Several different ROD Crate configurations are stored in the database(s) concurrently. At initialisation of a run, one configuration is selected and loaded.

The ROD Crate control takes all necessary actions required to fully control all modules of the ROD Crate and the RCW(s). It is based on the run control of the Online system and implemented as a tree of run controllers, one per ROD Crate and others on the RCW(s) as necessary. The ROD Crate controller (RCC) drives all RCMs of the ROD Crate into well-known states and investigates their status. It may require interaction with the TTC system and / or DCS.

The primary event data flow of the ROD Crate transports event data from the FE over the FEL, the ROD and the ROL to the ROS. The secondary event data flow of the ROD Crate transports sampled event data from the RODs over VMEbus, optional ROD Crate data collection, optional ROD Crate event building, and ROL to the ROS, or optionally to data storage for recording. ROD Crate DAQ provides collection of sampled event data from multiple RODs in the same ROD Crate and building of sampled event data from multiple ROD Crates.

Some ROD prototypes are not fully functional RODs and some are non-ROD modules, in particular at test beam, have to be read out at the same hierarchical level in the event data flow as a ROD. ROD emulation provides the missing functionality. ROD emulation may be based on the primary or on the secondary event data flow. In both cases, an RCP is required.

Monitoring is another basic function of the ROD Crate DAQ. Different types of monitoring have to be distinguished depending on the different types of monitoring data they are collecting. Event data monitoring provides event data coming from the secondary data flow. Scaler and histogram monitoring provides scaler and histograms derived from event data. Operational monitoring reads operational values not directly derived from event data.

ROD Crate calibration provides sub-detector calibration at the level of the ROD Crate. It reads all event data from the secondary event data flow, processes them and calculates calibration data. The calibration data are written to data storage for recording or to the calibration database.

ROD Crate event building is achieved by event building sources, one for each ROD Crate which participates in the event building, and one event building destination. An event building source is an output of a ROD emulation, monitoring or calibration activity. It sends all event data of the secondary event data flow over Local Area Network (LAN) to the event building destination. The event building destination is the input of a dedicated ROD emulation, monitoring or calibration activity and usually runs on the RCW.

The basic functions of ROD Crate DAQ can be combined to provide high-level functionality for physics and calibration runs. They can also be used in different setups for physics data taking, ROD emulation, event building from multiple ROD crates, and small laboratory setups.

The framework of ROD Crate DAQ is organized into four layers: hardware, operating system, low-level services, and high-level tasks. A call for tender for the hardware of the RCP is under way. It is assumed that PCs will be used for the hardware of the RCWs. Linux is the first choice of operating system. LynxOS will be used for the RCPs in case real-time performance is required. The low-level services, like libraries and drivers, are organized into three different layers for hardware access, high-level task support and support for the Online Software.

A ROD Crate DAQ task is a high-level task for ROD Crate controller, ROD emulation, monitoring, calibration or event building. ROD Crate DAQ tasks are provided as skeletons made of generic functions which may be extended by the sub-detector groups. Some standard functions are provided, e.g. for event building, which probably do not require extension.

The generic functions of the ROD Crate dataflow task are: the “input function” which reads data from FEL, RCM or LAN, the “processing function” which processes and selects data, the “output function” which sends data over the ROL or LAN, or to data storage, the “control function” which communicates with the ROD Crate controller, and the “monitoring/histogramming function” which communicates with the monitoring/histogramming of the Online Software. The specific tasks for ROD emulation, monitoring, calibration or event building are distinguished by the implementation of their individual functions

8.2.3.2 Implementation

ROD Crate DAQ re-uses existing software where possible. The Online Software is used as is, with some adaptation to sub-detector specific needs, in particular, for configuration. The ROD Crate controller is adapted from the controller developed for the ROS. The ROS software is also used to provide skeletons for the different tasks of ROD emulation, monitoring, calibration and event building.

The initial software development involves several real users and ROD Crates containing ROD prototypes as well as fully functional RODs. The workplan [8-18] allows for a first distribution of ROD Crate DAQ to be available by June 2003.

8.3 Boundary and interface to the level 1 trigger

Because ATLAS depends on data collection guided by RoIs the level 2 system needs information from the level 1 trigger decision. This information includes both the triggers which passed and the details of where, in eta and phi, the trigger primitives that caused the accept came from. This requires information internal to the level 1 system to be passed on to the HLT. Collecting this information and passing it on to the HLT is the responsibility of the RoIB.

Figure 8-14 shows the RoIB and its connections to the level 1 system. Since the level 1 accept rate is fairly high as an input transaction rate for a single processor, the RoIB is designed to spread the level 1 events over a small farm of processors which are referred to as supervisor processors. The supervisor processors receive a single S-link record containing the summary information for each event from the RoIB. Since the RoIB sends complete records to several supervisor processors no single processor has to deal with a full level 1 rate. The supervisors pass the level 1 data on to the level 2 processor that will make a decision on the event. The supervisors are responsible for a rudimentary form of load levelling. They are aware of the disposition of events that they send to level 2 processors and need to make sure that events are dealt with in a timely way. They also need to be assured that no single processor is overloaded with pending events. The operation of the supervisors is described in the Section 9.2.3.

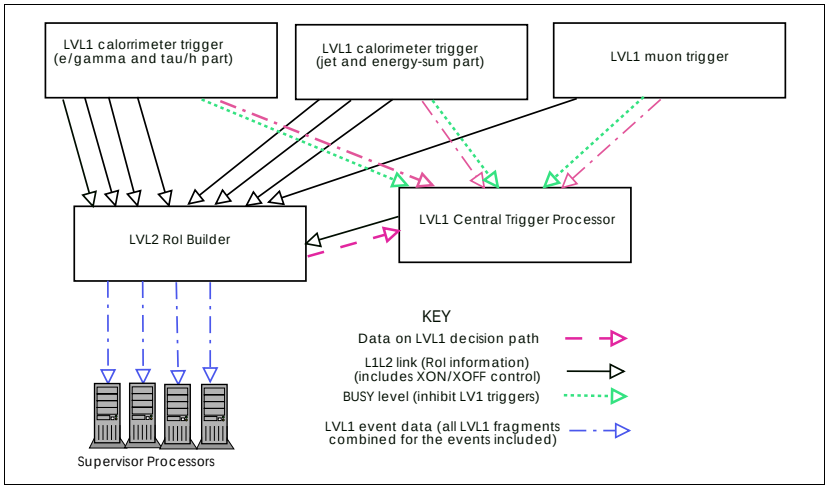


Figure 8-14 [reb001v000_lvl1lvl2.eps]

8.3.1 Description

This sub section should be a summary of what is detailed in [8-27].

A block diagram showing the level 1 system and its interconnection with the RoIB is shown in Figure 8-15. Each link from the level 1 system is an independent S-link input that sends a compact description of the event for that component. Each link is limited to sixty three 32 bit words or less per event. The various pieces of an event (referred to as fragments) will all arrive at the RoIB within one millisecond of each other. The RoIB will assemble the event data and send it to a supervisor processor which will then initiate the level 2 processing by passing a record to a level 2 processor which includes the pertinent level 1 RoI data.

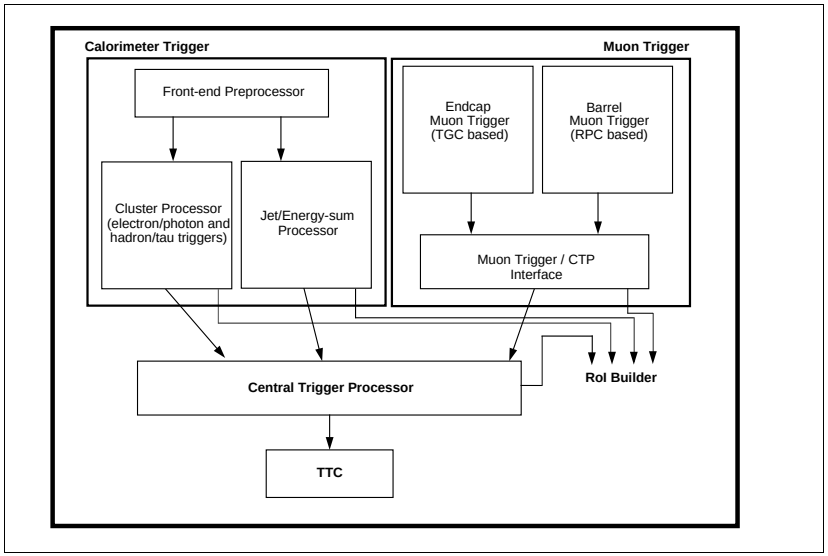


Figure 8-15 [reb001v000_lvl1block.eps]

8.3.2 Region of interest builder

There is overlap with Chapter 9, "High-level trigger" on this component and only one chapter should describe it in detail with the other just mentioning the specifics for that chapter.

The RoIB is a VME based system which uses FPGAs to combine the level 1 fragments into a single record. It is composed of two parts. A pair of cards buffer the level 1 input and direct fragments to cards which assemble individual events. Twelve inputs are considered adequate. This will include both the level 1 fragments and an independent TTC input to assure consistency between level 2 and the read-out system. The input cards will communicate over a dedicated backplane connection to one or more ‘builder’ cards that provide four outputs for assembled events. Figure 8-16 shows the system organization. The system can service four supervisors with a single ‘builder’ card and can be expanded in units of four by adding ‘builder’ cards.

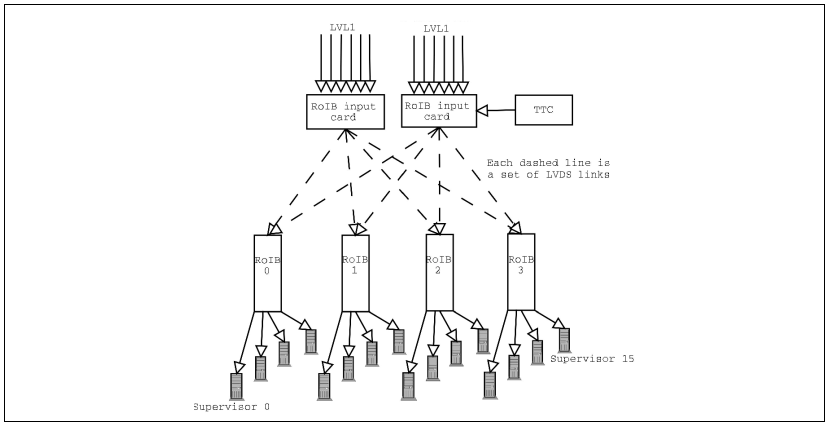


Figure 8-16 RoIB System organisation.

8.3.2.1 Detailed design

This sub section should expand on the High level design described in Section 5.5. It should be a summary of what is detailed in [8-27].

8.3.2.2 Performance

Based on results with (12 U) prototype, including results of integration studies with Level 1.

A prototype of the RoIB was fabricated and tested in 1999. This version was built using a pair of 12U VME cards, an input card capable of handling six S-link inputs and a pair of builder cards able to output to a pair of processors. This system utilized 76 Altera 10K40 FPGA’s and 8 10K50’s. Figure 8-17 shows the pair of boards that were built. The system and early performance measurements are documented in [8-22].

This system was adequate to demonstrate a number of critical points. It showed that the idea of combining records from several sources using an FPGA based device is feasible. It showed that the communications overhead for processors would not result in unmanageable numbers of processors just to handle the 100kHz event rate; four 300 MHz pentium two machines were adequate to handle the 100kHz rate. Subsequent tests with several prototype pieces of the level 1 system (the muon-CTP interface and the calorimeter CPROD) made a start on debugging the component interfaces and further demonstrated that external inputs could be handled at the expected rates [8-23].

The scale of a full system will need to be set in the future, but current indications from the early prototype make it clear that the full system will involve the number of processors needed to satisfy the HLT functions of the supervisors documented in the HLT description and testing section plus a few processors (less than four) to cover the additional communications from the RoIB.

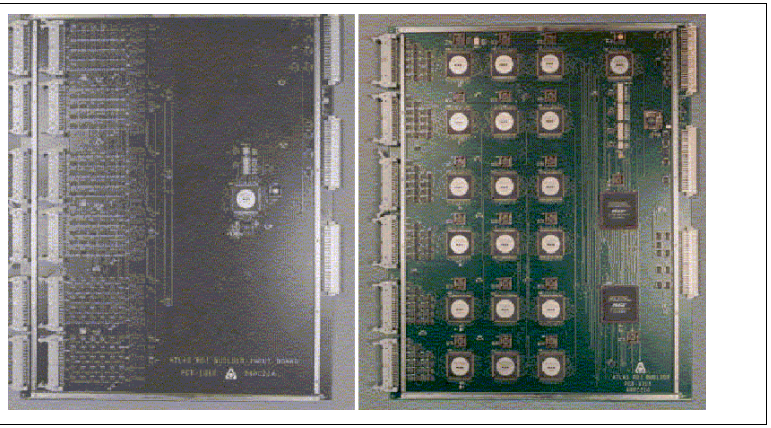


Figure 8-17 12U VME card prototype of the RoIB.

8.4 Control and flow of event data to high level triggers

8.4.1 Message passing

8.4.1.1 Control and event data messages

Introduce the types of messages, the flow of messages, message rates and the bandwidths required. Concluding with the choice of link technology.

The flow of event data from the ROS, where data are buffered during the LVL2 and event building latencies, to the HLT is achieved by the exchange of control messages and subsequent event data messages between components of the DataFlow system. This is described in detail in [8-24] and here only its major features are summarized. Figure 8-18 shows a sequence diagram detailing the base interactions between DataFlow components.

8.4.1.1.1 L2SV

The LVL2 Supervisor receives LVL1 Results containing the RoI information from the RoI Builder. It assigns according to a load balancing algorithm a L2PU to analyse the event. It will then receive the LVL2 decision from the L2PU which it forwards to the DFM. In case no LVL2 decision will be received within a pre-defined timeout (e.g. the L2PU crashed, or a message was lost), the L2SV will treat the event as if accepted by the L2PU. There will be several L2SV deployed in the final system.

A LVL1 Result message does not exceed 512 bytes [8-27], leading to a maximum bandwidth requirement of O(50 MB/s) at 100 kHz LVL1 rate. This bandwidth can be handled easily with any

Figure 8-18 Sequence diagram showing the interactions between DataFlow components

gigabit capable link technology, furthermore, it has to be divided by the number of L2SVs deployed.

8.4.1.1.2 2.2 L2PU

A LVL2 Processing Units receives a LVL1 result containing RoI Information from the L2SV, which it uses to seed its processing algorithms. It requests specific RoI data from selected ROSs for analysis, more RoI data is requested repeatedly until the sequential processing results in its final decision to either reject or accept the event.

This includes also the decision whether an accepted event should be prescaled, or a rejected event should become forced-accept. The L2PU then sends the LVL2 decision (accept, reject, prescaled, forced-accept) back to the L2SV and in case of an (forced-) accepted event it also sends a detailed record to the pseudo-ROS. There will be many hundreds of L2PUs deployed in the final system.

As any individual L2PU processes events at a rate not higher than O(100 Hz), the bandwidth requirement for receiving the LVL1 results are O(1 kB/s). However, the data rate of the received RoI data can be substantial and is estimated to reach maximum values as high as O(10-20 MB/s).

8.4.1.1.3 ROS

The Read-out System holds event data fragments inside Read-out Buffers (ROBs) and makes them available to HLT, following data requests. The ROBs are cleared on receipt of clear messages.

The maximum bandwidth out of an individual ROB is estimated to O(10 MB/s) and data fragments are requested at up to xxx kHz [8-25]. Depending on how many ROBs are hold in a ROS and of the access mode to the ROBs, the ROS needs to provide O(Y MB/s) at a rate of yyy kHz.

8.4.1.1.4 2.5 pROS

The Pseudo-ROS receives the detailed result records of the L2PUs for accepted events and participates to the event building process, such that the LVL2 detailed result appears within the full event record. From the point of view of the SFI there is no difference between the pROS and a normal ROS component. However, the pROS has no detector specific input at all and requires no special hardware like e.g. a ROBIn.

Given a LVL2 accept rate of O(2 kHz) and an estimated size of the LVL2 detailed result record of O(1 kByte), the bandwidth in and out of the pROS will be a O(2 MB/s). One pseudo-ROS will therefore be sufficient for the final system.

8.4.1.1.5 2.6 DFM

The Dataflow manager receives (grouped) LVL2 decisions from the L2SVs and assigns an SFI, following a load balancing algorithm, for the event building of every accepted event. It multicasts the (grouped) clear messages to all ROSs (incl. pROS).

The bandwidth requirements for the exchange of control messages with the DFM are small, given the grouping factor of the LVL2 decisions is O(100) results at a LVL1 rate of 100 kHz to O(1 kHz) for the reception of the LVL2 decision messages. The communication with the SFI adds an additional two times O(2 kHz) message rate. The distribution of the clear messages from the DFM to the ROSs will also be grouped. with an assumed grouping factor of O(300), only O(300 Hz) of message rate will need to be added - given a multicast mechanism for the distribution of clear messages. The total message rate to be handled by the DFM is therefore O(6 kHz); only small messages O (few 100 Bytes) need to be exchanged, leading to an aggregated bandwidth requirement of O(3 MB/s) to be handled by the DFM.

8.4.1.1.6 SFI

The SubFarm Input assembles event fragments from the ROSs (incl. pROS) and serves these to EventFilter SubFarms. It is the event building component in the DataFlow system and has to stand relatively high data and control rates, and conversely the bandwidth (for the data). The event size for complete events is O(2 MB) and the event building rate O(2 kHz), resulting into an aggregated bandwidth requirement of O(4 GB/s). This load needs to be distributed over many SFIs. Assuming an event building rate of O(50 MB/s) handled by one SFI, O(80) SFIs, each building events at O(40 Hz), will need to be deployed in the final system. The message rate to be handled by an SFI depends on the number of ROSs deployed, and is O(40 Hz) times two times the number of ROSs. It will not be above O(64 kHz) in case of 1600 ROS deployed.

For the communication with the EventFilter SubFarms, only a few messages need to be exchanged per event. However, a bulk transfer of the full event record is needed.

None of the afore mentioned messages require rates and bandwidth which cannot be handled by a wide range of link technologies. A commodity solution of widely available products on the world market can be deployed. Here the choice is dictated by price, long term availability, support, inter-operability and suitability for ATLAS DataFlow. Ethernet in its varieties of 100 Mb/s and 1000 Mb/s is the prime candidate and has been evaluated to prove its suitability for exchange of control and event data messages in the Atlas DataFlow.

8.4.1.2 Ethernet

This section should introduce the key features (i.e. VLANS, QoS, switches, flow control) supporting its selection and how they will be used. Should also summarise, based on [8-29], the basic message passing capabilities in terms of achieved rates, overheads and CPU loads.

8.4.1.3 Design of the message passing component

Presents the main features of the design (high Level enough?) based on [8-30].

8.4.1.4 Performance of the message passing

Presents, based on [8-29], the performance of the message passing component in terms of achieved rates, overheads and CPU loads.

The message passing layer of the data flow software is responsible for the transfer of all control and event data between different components. It provides a common technology-independent API across all applications.

The message passing layer itself imposes no structure on the data which is exchanged. Rather, this structure is defined by the message types in [see Section 8.4.1.1] which can be changed without affecting the message passing per se.

The service it provides is the transfer of up to 64 kByte of data with only a best-effort guarantee. No re-transmission or acknowledgement of data is done by this layer. This allows to implement the API over a wide range of technologies without imposing un-necessary overhead where not needed or duplicating existing functionality. The API supports the sending of both unicast and multicast messages. The latter has to be emulated by the implementation if it is not available (e.g. for TCP).

The message passing layer interface has been implemented over raw ethernet frames, UDP and TCP. The latter two implementations are technology-independent per se, although systematic measurements were only done for switched ethernet configurations (i.e. the routing aspects of IP are not necessary for the ATLAS architecture). TCP provides additional reliability compared to UDP and raw ethernet. However, applications and message flow have been designed in such a way that the system will still work when running over an unreliable technology. The raw ethernet implementation adds message re-assembly on the receiver side, similar to what IP provides. Otherwise the maximum message size would be restricted to a single ethernet frame which was seen as too restrictive for the range of data sizes intended.

Internally all implementations support scatter / gather transmission and reception of data. This allows to build a logical message out of a message header and additional user data that doesn't need to be copied inside the application.

The basic operations of allocating and de-allocating a buffer to send or receive are dominated by the need to make the interface thread-safe. On a 1 GHz dual processor SMP machine they take in the order of 0.6 µs each. Since they require main memory access on a real multiprocessor system this does not scale with the CPU frequency but with the memory speed. On a 2.2 GHz machine the numbers are only slightly smaller.

The basic measurements can be compared to the direct socket measurements of [see Section 8.4.1.2]. Note that the latter are done in a single-threaded environment without any dynamic memory allocation and always send and receive data from a fixed location and with a fixed size that is known in advance. Raw ethernet measurements are only done with a maximum of 1460 bytes, since no re-assembly of larger packets has been implemented. They therefore provide an upper bound for the possible performance which we don't expect to reach with the additional functionality in the message passing layer.

Among the reasons for lower performance are:

- Support for scatter / gather operations requires the kernel to copy an additional user data structure across kernel boundaries.
- The varying message length supported by the message passing API requires at least two read system calls for TCP, and two read system calls for every raw ethernet packet [check with David Botterrill, maybe just for first packet].
- For raw ethernet the re-assembly of frames into large messages.
- Thread-safety for the sender side: multiple threads can send at the same time as long as their destinations differ. They are serialized when they both send to the same destination.

Problems with scalability are only expected for the TCP implementation where a potentially large number of open sockets has be handled. The default TCP code uses the select() system call which is known not to scale. However, alternatives are available either in the form of POSIX conforming real-time signal in combination with non-blocking sockets or in a non-standard form by ongoing developments in the latest Linux kernel (epoll() interface[ref]). The UDP and raw ethernet implementations use only a single socket for receiving data.

Repeating the measurements for request / response and streaming shows an overall overhead of about 10 µs compared to the low-level tests. This translates into a time of about 22 µs to serve an incoming message or a rate of 45.5 kHz for receiving. These numbers are for a dual 2.2 GHz machine and require proper settings of socket buffer space and interrupt coalescence for the driver. E.g. with default settings for the interrupt coalescence this rate drops to 7 kHz. The difference between the various implementations are negligible: they are about 22.9 µs for UDP and 21.09 µs for raw ethernet.

Finally we can compare the numbers measured above with the observed rates of one of the applications in the data flow. The SFI application can do event building with raw ethernet frames at an overall rate that corresponds to a 40 kHz of data packages (ca. 1400 bytes) input and 40 kHz of requests (<64 bytes) output rate [8-32] which is in overall alignment with the low-level measurements.

8.4.2 Data collection

8.4.2.1 General overview

This section describes the common model to collecting data for level 2 processing and event building.

DataCollection is a subsystem of the Atlas TDAQ DataFlow system responsible for the movement of event data from the ROS to the LVL2 Processing and to the EventFilter and also to MassStorage. See.

It includes the movement of the LVL1 RoIs to the LVL2 PU (via the LVL2 SuperVisor) and the LVL2 result (decision and detailed result) to the EventFilter as well as the EventBuilding and feeding the complete events to the EventFilter.

However, DataCollection is not responsible for initializing and formatting (or preprocessing) of event fragments inside the ROS, neither is it responsible to do preprocessing nor to perform trigger decisions in the LVL2 Processing Unit or in the EF SubFarm.

Figure 8-19 shows a context diagram of the two main components of DataCollection (LVL2 DataCollection and EventBuilding) and its interfaces to other systems and subsystems of Atlas TDAQ.

Figure 8-19 DataCollection context diagram.

The following lists the applications to be provided by DataCollection:

L2SV LVL2 SuperVisor

L2PUA	LVL2 Processing Unit Application (i.e. L2PU low layer functionality)
DFM	DataFlow Manager
pROS	Pseudo ROS
SFI	SubFarm Input
SFO	SubFarm Output

In order to deploy this variety of components, a common approach in design and implementation is envisaged. This approach lead to the definition of the common DataCollection framework, implementing a suite of common services. These were found to be:

- OS Abstraction Layer
- Configuration Database
- Error Reporting
- System Monitoring
- Run Control
- Message Passing

All applications in the DataCollection software share the need for a common set of typical operations. This includes error logging, configuration, system monitoring, run control and message passing. All these capabilities are provided by a set of packages which is usually referred to as the DataCollection Application Framework. This design leads to a large code reuse in practice. A typical application is built on top of a skeleton application and only has to provide the actual additional functionality.

Services are built from packages following a modular approach. Many of these packages consist of interfaces only, whose implementation is provided by other packages which can be changed at configuration or run-time. Examples are the error reporting (switching between simple stdout/stderr and MRS), the configuration database (switching between OKS files and remote database server), the system monitoring (providing an interface to the Information Service of the Online Software and a local independent version). The message passing interface allows the concurrent existence of multiple implementations at the same time. E.g. all of UDP, TCP and raw ethernet sockets can be used by a single application in a given setup.

This clear separation between interfaces and implementations exists down to the lowest levels like the thread interface and access to clocks and timers.

8.4.2.1.1 OS Abstraction Layer

The OS abstraction layer consist of packages hiding all OS specific interfaces. E.g. the *threads* package hides the details of the underlying POSIX thread interface.

8.4.2.1.2 Error Reporting

The ErrorReporting package allows to log error messages either to stdout/stderr or to MRS. Each package can define its own set of error messages and error codes. Error logging can be enabled/disabled on a package by package basis, with a separate debug and error level for each package. Furthermore debug logs and normal error logs are treated logically differently, so the

debug message could go to stderr while all normal application logs go to MRS. The user only interfaces via a set of macros to the ErrorReporting system. This allows to compile out the debug macros for optimized builds.

8.4.2.1.3 Configuration Database

All applications make use of the Online Software configuration database through the API provided by these packages. The design uses the Bridge pattern described in Gamma et al. This allows to change the underlying implementation without the client code noticing it.

The user's view of the database is hidden by configuration objects, which read the database and provide a more convenient way to access the information. Database schema file evolutions are coped with an automated re-creation of the C++ code for these configuration objects out of the schema file only

8.4.2.1.4 System Monitoring

This package allows every component to make arbitrary information available to some outside client. In practice this is used to publish statistics like counters and histograms. Users inherit from the *Resource* class and implement a virtual function. The packages makes this information available in various different ways, including the Information Service of the Online Software.

Again the interface is strictly separated from the different implementations, so users are unaware of it and the implementation can change without them noticing it.

8.4.2.1.5 Run Control

The run control interface is responsible for translating the requests from the Online Software about state changes into commands for the application. It also provides a skeleton around which one can build an application.

These classes realize most of the use cases for run control. They talk to a special DataCollection Run Controller on the one side and to user code on the other side.

8.4.2.1.6 Message Passing

The Message Passing Layer defines a couple of classes to allow the sending and receiving of messages. The *Node, Group* and *Address* classes are used at configuration time to setup all the necessary internal connections.

The *Port* class is the central interface for sending data. All user data has be in part of a Buffer object to send or receive it. The *Buffer* interface allows to add user defined areas which are not under the control of the Message Passing layer to avoid copying.

The *Provider* class is an internal interface from which different implementations have to inherit. Multiple *Provider* objects can be active at any given time. A *Provider* is basically the code to send and receive data over a given protocol/technology, e.g. TCP, UDP or raw ethernet.

Using the DataCollection Application Framework, the DataCollection components were implemented efficiently with maximum reuse of code and coherency in system aspects.

The interaction of the DataCollection components is detailed in the following to sections for LVL2 DataCollection and event building.

8.4.2.2 RoI data collection

8.4.2.2.1 Design

This section should describe the interaction between applications which results in the collection of data at the level 2 processing unit.

8.4.2.2.2 Performance

8.4.2.3 Event Building

8.4.2.3.1 Design

This section should describe the interaction between applications which results in the collection of event fragments to form a complete event at the SFI. Should also include the aspects related to traffic shaping.

8.4.2.3.2 Performance

Text still to be provided

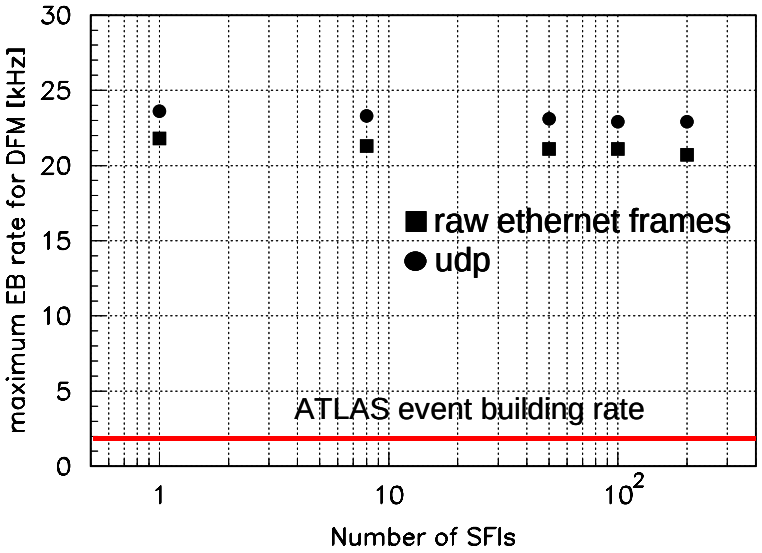


Figure 8-20 Event Building 1.

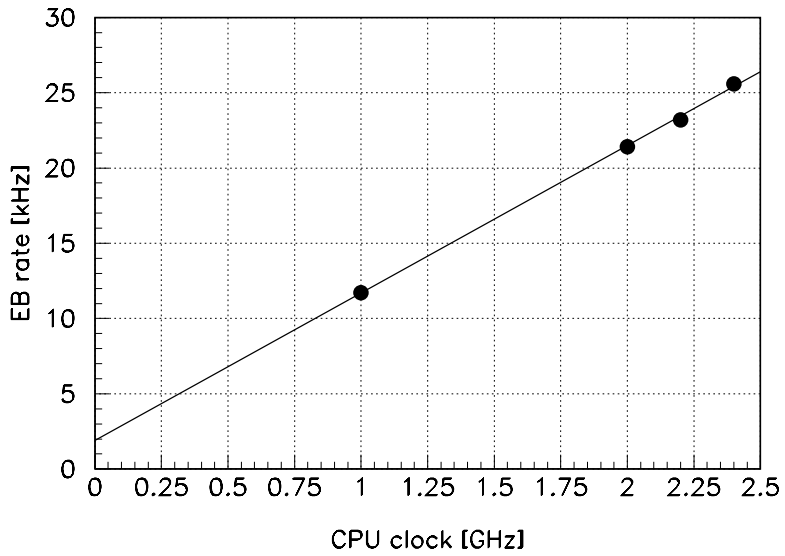


Figure 8-21 Event Building 2.

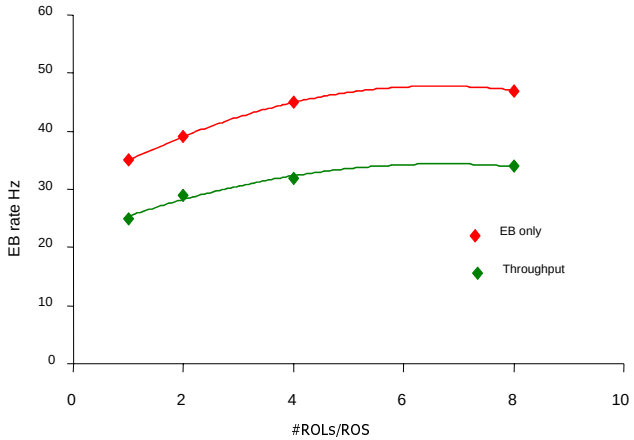


Figure 8-22 Event Building 3.

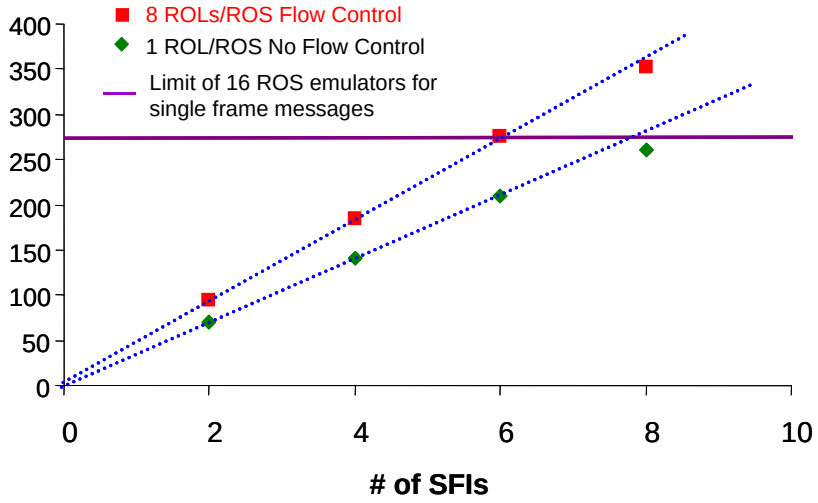


Figure 8-23 8 ROLs/ROS 4.

8.5 Reliability and fault tolerance: this section has been moved to Chapter 6, "Fault Tolerance and Error Handling"

8.5 Configuration, control and operational monitoring

The sub-sections in this section have been moved to other chapters.

Local Controller: moved to Chapter 12, "Experiment Control".

Configuration data: moved to Section 10.4, "Databases".

Operational monitoring: moved to Chapter 7, "Monitoring"

8.5 Scalability

8.5.1 Detector read-out channels

This section describes quantitatively how the physical size, performance and control and configuration of the system scales with the "amount" of detector to be read-out.

8.5.1.1 Control and flow of event data

How the number of applications, messages and data volume changes.

8.5.1.2 Configuration and control

Amount of configuration data a function of the amount of detector.

8.5.2 Level 1 rate

How the system performance and physical size scales with respect to the level 1 rate.

8.6 References

8-1 Trigger & DAQ Interfaces with Front-End Systems: Requirement Document http://atlasinfo.cern.ch/Atlas/GROUPS/DAQTRIG/DIG/archive/document/FEdoc_2.5.pdf

8-2 ATLAS High-level Triggers, DAQ and DCS Technical Proposal, CERN/LHCC/2000-17, 31 March 2000.http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/SG/TP/tp_doc.html

8-3 The S-LINK Interface Specification. http://edmsoraweb.cern.ch:8001/cedar/doc.info?document_id=110828

8-4 The raw event format, http://atlasinfo.cern.ch/Atlas/GROUPS/DAQTRIG/ROS/DAQ-1_Note_050_update_1a.pdf

8-5 Recommendations of the Detector Interface Group - ROD Working Group https://edms.cern.ch/document/332389/1

8-6 The CMC standard. Common Mezzanine Cards as defined in IEEE P1386/Draft 2.0 04-APR-1995, Standard for a Common Mezzanine Card Family: CMC (the CMC Standard).

8-7 Design specification for HOLA, https://edms.cern.ch/document/330901/1

8-8 Procedures for Standalone ROD-ROL Testing, G. Lehmann *et. al.*, 27 July 2001, ATC-TD-TP-0001 http://edmsoraweb.cern.ch:8001/cedardoc.info?document_id=320873&version=1

8-9 ROS URD

8-10 Read out system high level design

8-11 ROS Local Controller

8-12 ROBIN Summary document

8-13 Readout sub-system test report (using DAQ -1.

8-14 ROBIN HLDD

8-15 ROBIN DLDD

8-16 ROBIN SWID

8-17 ROD crate DAQ design

8-18 ROD crate DAQ workplan

8-19 Reference to the RACE board

8-20 ROS Test Report

8-21 Reference to FPGA emulators

8-22 ATL-DAQ-99-016

8-23 cite ATL-DA-ER-0016 & the corresponding muon-CTP i/f document

8-24 Data collection note # 012... to be moved into EDMS

8-25 Paper model results

8-26 Data Collection URD

8-27 Level 1 - Level 2 interface document

8-28 RoI Builder URD

8-29 Results of basic comms tests

8-30 Design of the message passing component

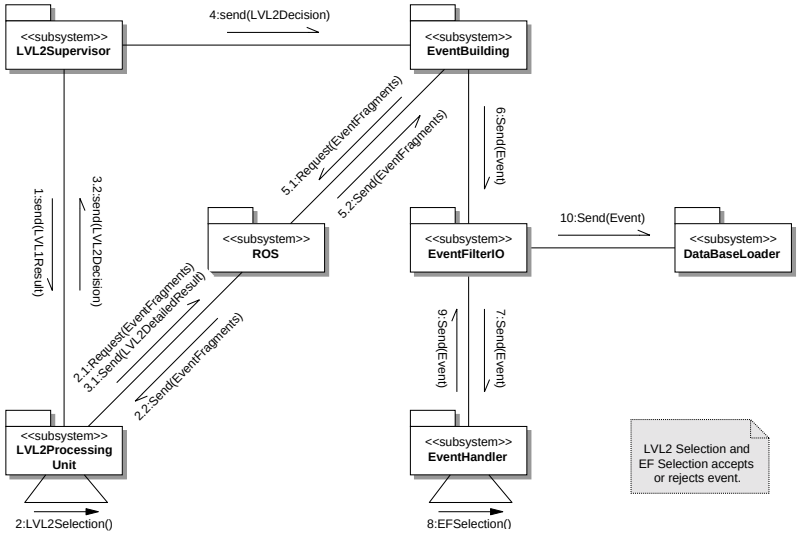
8-31 Documents supporting technology choices

8-32 DataCollection test report

8-33 DataCollection Local Controller

9 High-level trigger

9.1 HLT Overview



The figure shows how the subsystems collaborate by an exchange of messages. In the online system the LVL2 selection is done in the LVL2 processing unit and the EF selection in the event handler. Both, LVL2 and EF, are situated in dedicated processor farms. The communication of the LVL2 processing unit is different in nature from the communication of the event handler. LVL2 receives the LVL1 result from the LVL2 supervisor. LVL2 is ROI guided and only requests the corresponding fragments of the events from the ROS. The data is read out of the read out buffers (ROBs), which hold the event data after the LVL1 accept. After a positive LVL2 decision the event building collects all fragments, including the LVL2 result. The full event is sent via the event filter IO to the event handler, where the EF selection is made. Accepted events are sent to the data base loader for permanent storage of the event for offline reconstruction and analysis.

9.2 Level 2

9.2.1 Overview

Includes use RoI mechanism (*i.e. selective Read-out*), requirements and interplay between components.

9.2.2 RoI Builder

There is overlap with Chapter 8, "Data-flow" on this component and only one chapter should describe it in detail with the other just mentioning the specifics for that chapter.

Main description of design, implementation, interfaces etc should be in Chapter 8 (DataFlow). Here limited to a brief description of the functions (i.e. gathering together of the LVL1 RoI information and then routing to a Supervisor processor)

9.2.3 LVL2 Supervisor

Main description of functions, design and implementation here in this chapter, should refer back to the DataCollection Framework described in Chapter 8. The description here to include the load balancing aspects. (The DataFlow chapter should be limited to a description of function of the Supervisor in DataFlow and the rate at which it handles messages - i.e. the performance measurements (30 kHz rate).)

9.2.4 LVL2 Processors

Need a description of how the software inside a LVL2 processor is structured. i.e. L2PU hosting the PSC, which hosts the event selection code. Also how the algorithms access data from the ROB's with the interface layers provided between the algorithm and the L2PU. Diagrams to be included here are ones showing the multi-threaded nature of the Worker threads and the sequence diagram of what happens during configuration and in the event loop.

Should include a statement about the possible use of FPGA's here with a reference to the FPGA implementation back-up document, but note that this is not included in the baseline option.

9.2.4.1 L2PU

The design and implementation of the L2PU is based on the DataCollection Framework described in xyz from which it uses the following services: application control, initialisation & configuration, error reporting, application monitoring, message passing, and, for the purpose of performance evaluation, the instrumentation.

The L2PU communicates with the LVL2 Supervisor from which it receives the RoI information (originating from the LVL1 Trigger) and to which it returns the LVL2 Trigger decision. RoI Data (in the form of RoB fragments) is requested from the ROSs, on instigation of the LVL2 Selection algorithms. For positive decisions, a LVL2 Result is sent to the pROS.

The actual selection algorithms runs inside one out of several 'Workerthreads', each processing one event. This multi-threaded approach has been chosen to avoid stalling the CPU when waiting for requested RoI data to arrive (from ROSs). This also allows efficient use of multi-CPU processors but LVL2 selection algorithms must be thread-safe. Specific guidelines to developers are given in [9-7]. Some asynchronous services (application monitoring, input of data) are also executed in separate threads.

The 'RobDataCollector' is a service that takes a 'list of RoBs' as input parameter and returns a 'list of RoB fragments'. The RobdataCollector takes care of sending out requests for data to the

appropriate ROSs, waits for all data to arrive, assembles the received ROS fragments into a list of RoB fragments which are returned to the caller.

The LVL2 event selection takes place inside the PSC (PESA Steering Controller) which has a simple interface to the DataCollection framework: it receives the LVL1 RoI information ('LVL1 Result') as input parameter and it returns the LVL2 result. Selection algorithms that need RoI data activate the 'RobDataCollector'.

Figure 9-1 illustrates what happens for each event. The LVL2 Supervisor selects the L2PU with the smallest number of outstanding events and sends the LVL1 RoI information. This 'schedules' the L2PU which stores the received LVL1 Result in a shared queue. Any available Worker-thread unqueues the event, starts processing it, derives a LVL1 Decision from the LVL2 Result which is returned to the LVL2 Supervisor. For positive decisions, the LVL2 Result is also sent to the pROS.

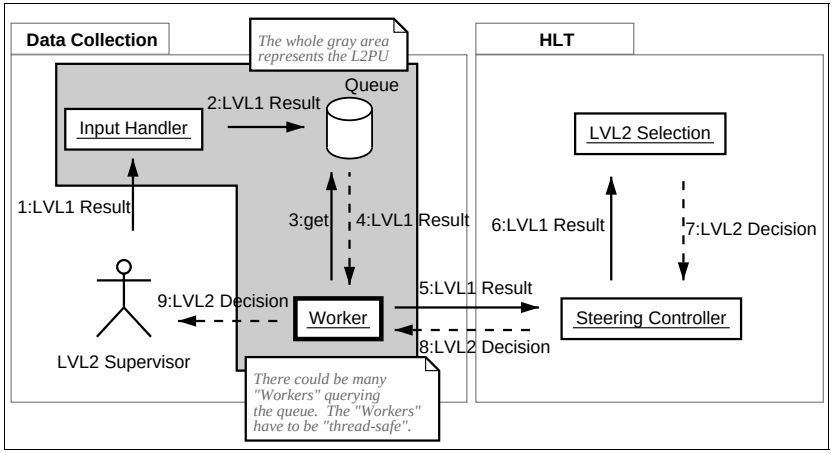


Figure 9-1 Collaboration diagram showing the successive steps that are applied to each event received from LVL1 leading to the LVL2 Decision.

The collection of RoB data is shown in Figure 9-2. The DataCollector is takes a 'list of RoBs' as input parameter and returns a 'list of RoB Fragments'. The DataCollector takes care of sending out requests for data to the appropriate ROSs, waits for all data to arrive, assembles the received ROS fragments into a list of RoB fragments which are returned to the caller.

The LVL2 event selection takes place inside the PSC (PESA Steering Controller) which has a simple interface to the DataCollection framework: it receives the LVL1 RoI information ('LVL1 Result') as input parameter and it returns the LVL2 result. Selection algorithms that need RoI data activate the DataCollector. The PSC must be state aware and respond to run control commands of the L2PU.

The DataCollection performance has been measured in testbeds. It exceeds by a large margin the required I/O capacity. The performance is illustrated in Figure 9-3 (RoI building, scaling)..

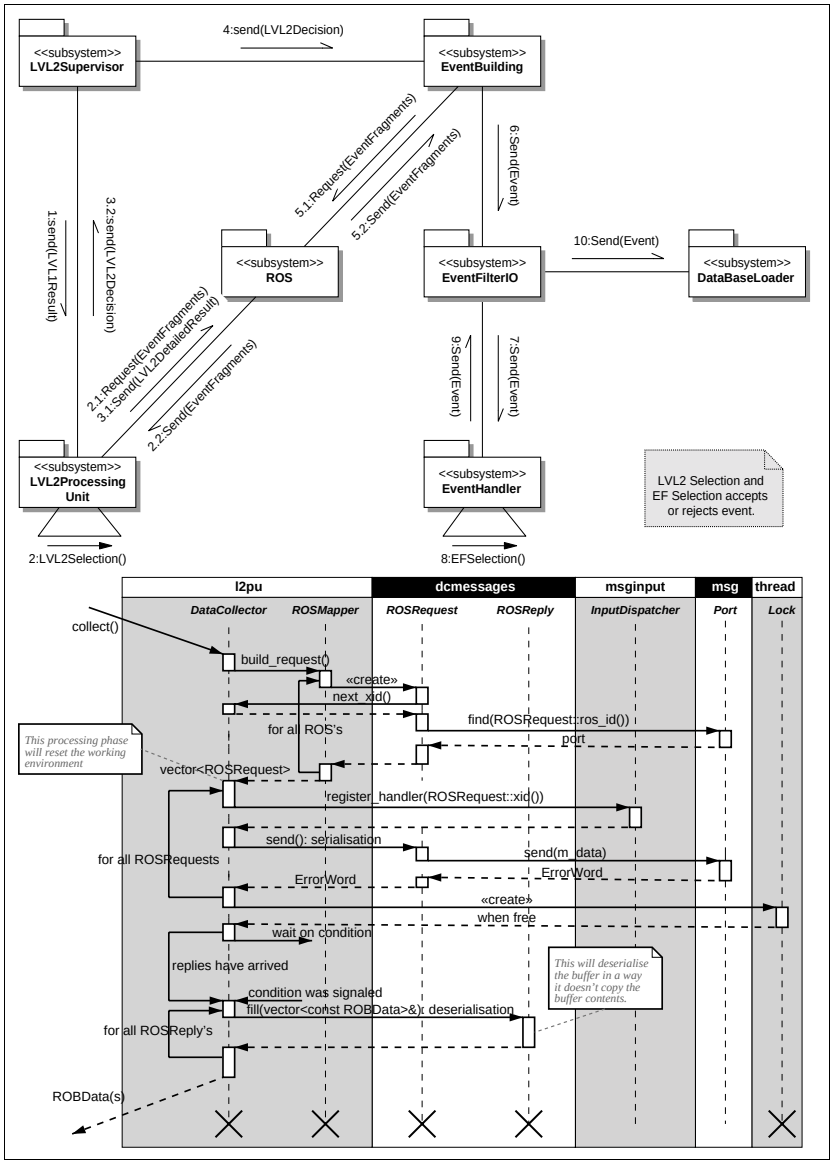


Figure 9-2 Data Collection. Provisional! The two diagrams need to be merged into a simpler (collaboration?) diagram with less technical detail.

Placeholder for a plot showing RoI Building Performance.

Figure 9-3 L2PU data Collection Performance.

9.2.4.2 PSC (PESA Steering Controller)

The PESA Steering Controller (PSC) is the HLT component that interfaces the L2PU and the LVL2 event selection software. The mission of the PSC is threefold: it allows the L2PU to host and control selection software developed in the offline framework; it allows the algorithm steering software to be shared with the Event Filter; and it provides a mechanism for transmitting the LVL1 and LVL2 results between the dataflow system and the PESA software.

The key to the PSC design is to place this interface where the functionality of the dataflow and event selection frameworks can be cleanly separated. One such location is the Finite State Machine (FSM) of the L2PU. The PSC can then be realized as a local “state-aware” replica of the Data Collection’s FSM. It thus provides the means for forwarding state changes from the dataflow software to the PESA software. Since the HLT event selection software is being developed in the offline framework Athena [9-8], which is itself based on Gaudi [9-9], the PSC has been designed [9-10] to re-use the framework interfaces defined in Gaudi.

Figure 9-4 illustrates the sequence of interactions of the PSC with the dataflow and the PESA software. The figure shows three states: *Configure*, *Start*, and *Stop*. During the *Configure* phase, configuration and conditions metadata is obtained from external databases via an HLT-online interface. These data are then used to configure the PESA software and all associated components. As Figure 9-4 (left) shows, during this phase multiple Worker Threads are also set up. After a *Start*, the PSC receives an ‘execute event’ directive with a LVL1 result as an argument. The PSC then returns (after execution of the PESA selection software on the event) the LVL2 result directly to the Data Collection framework.

An important aspect of this approach is that the LVL2 event handling is managed entirely by the Data Collection framework. The PSC then does not need to interact directly with the input thread, the Level-2 supervisor, or with the pROS. The requests for event data fragments are hidden behind the DataManager.

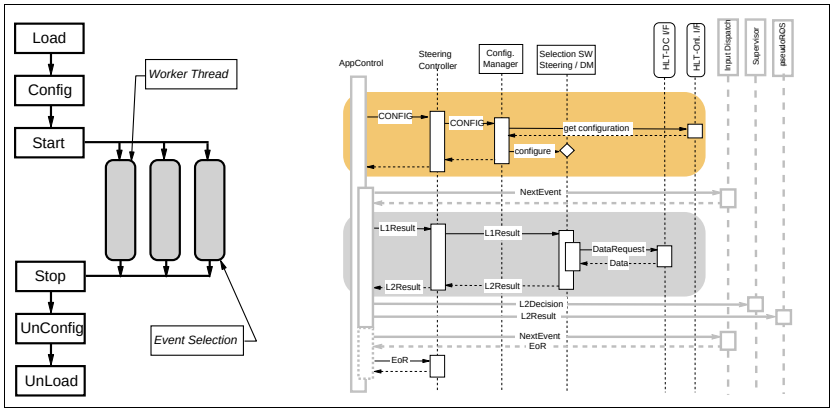


Figure 9-4 The L2PU Finite State Machine (left) and the PESA Steering Controller (right). Provisional, figure is being updated!

After a *Stop*, the PSC terminates algorithm execution. At this stage, run summary information can be produced for the selection process.

Since the event selection software executes in multiple worker threads, the PSC must provide a thread-safe environment. At the same time, and in order to provide an easy-to-use framework for offline developers, the PSC must hide all technical details of thread handling and locks. Thread safety has been implemented in the PSC by using Gaudi’s name-based object and service bookkeeping system. Copies of components that need to be thread-safe are created in each worker thread with different labels. The labels incorporate the thread-ID of the worker thread, as obtained from the Data Collection software. The number of threads created by the Data Collection software is transferred to the PSC, which transparently creates the number of required copies. In this scheme, the same configuration can be used in the offline and in the LVL2 environments; the thread-ID collapses to *null* in the offline software.

After integrating the PSC with the DataCollection software, both performance and robustness tests were carried out on a dual-processor 1.533 GHz Athlon machine (for details, see [9-10]). The PSC ran for over 50 hours with three threads with an early selection software prototype [9-11]. The prototype ran successfully on both single- and double-CPU machines, showing it to be thread safe. A direct measurement of the PSC overhead yielded 13 microseconds per event, well within the 10 millisecond nominal LVL2 budget.

9.2.4.3 Data access i/f’s

Access to data from the HLT software is explained in section 9.4 . The same interface is seen by Offline, EF and LVL2 but a specific implementation of the LVL2 ROBDataProvider interfaces to the LVL2 DataCollector as shown in Figure 9-5..

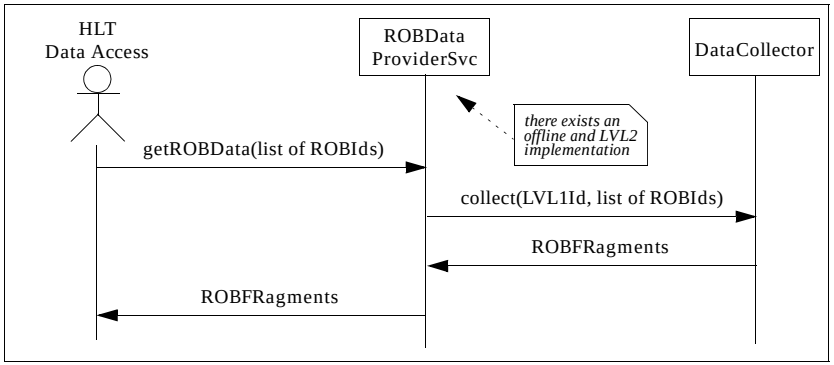


Figure 9-5 LVL2 ROBDataProvider. A LVL2 specific implementation of the ROBDataProvider Service provides a unique interface to the HLT Data Access to Offline, EF and LVL2 but interfaces to the LVL2 DataCollector.

9.2.5 pROS

Main description is in Chapter 8, "Data-flow". Here just a brief note to describe the function from an HLT perspective. i.e. The mechanism to receive the LVL2 result for inclusion in the built event - thus passing the result from LVL2 to the EF.

9.2.6 LVL2 Operation

Here a brief description of how the LVL2 processors are configured, controlled and monitored, how they are organised into sub-farms and how the farm-fabric is managed.

9.3 Event Filter

The Event Filter (EF) is the third and last step of the selection chain. It makes use of the full available information. It will use the offline framework (ATHENA) to execute filtering algorithms which will be taken directly from the offline suite.

9.3.1 Overview

9.3.1.1 Functionality

The functionality of the EF has been logically distributed between two main entities:

- • the Event Handler (EH) in charge of performing the activities related to event selection. This includes the data flow between the main DAQ system and the EF as well as between the different steps of the selection itself. It also includes the framework to run the processing tasks (PT).

- the Supervisor in charge of the control operations, in co-ordination with the overall TDAQ control system. Its responsibilities encompass also the monitoring of the EF functionality.

Some extra functionality may be possibly added to the EF. Examples of such extra activities are the global monitoring of the detectors or tasks related to alignment and calibration. Although those tasks have not yet been formally assigned to the EF, care has been taken so that they can be easily plugged in the EF context without jeopardising the selection activity.

9.3.1.2 Operational analysis

The operational analysis of the whole HLT has been described in a dedicated document [1] which describes in details the expected functionality and gives uses cases for the operation.

In the case of the EF, use cases are separated between the following sections:
***(Note**: each of these bullets could be developed to give more details on the operations)*

- the *start-up operations*, which tentatively gives the list of the different operations which should be necessary to set the HLT processing farms in operation.
- the *Run Control* related operations, which illustrates how the EF will cope with the *Start of Run*, *End of Run*, *Pause*, *Resume* and *Checkpoint* commands in their present understanding.
- the *shutdown* operations which tentatively gives the list of the different operations which should be necessary to set the HLT processing farms back into the non-existent state
- the *steady operation actions*, which are the list of the operations which should not imply to formally stop the data taking process, but are rather associated with the *checkpoint* procedure. This includes changing the trigger menu during a given spill as well as modifying the computing power of the EF (by adding or removing CPUs in the farm).
- *unsolicited* events: this section is a first approach to error management in the EF. More information on error handling in EF can be found in Chapter 6 of the present document.

9.3.2 Event Handler

A detailed list of requirements can be found in [2]. This list is based on the analysis of constraints coming from other systems and on some first and general uses cases. A summary of these requirements is given here.

- The EH shall receive events from the main Data Flow system and send them back to it to be kept in permanent storage. The EH shall possibly append information to the event containing details related to the performed selection operations. Events may be directed towards dedicated output channels according to the results of the performed processing (specialised channels according to event classification, monitoring channels, channels for events having led to error while processing, etc...)
- The EH shall distribute events to specific processing tasks according to a specific information contained in the event header. This requirement comes as an extra functionality in addition to the specialised distribution of events to dedicated sub-farms which can be performed by the Data Flow system.
- The EH shall provide the software infrastructure to perform the required treatment inside the Processing Tasks. Filtering is mandatory, as well as functionality for EF, monitoring purposes. Additional function-

ality for general monitoring, calibration check, etc... should also be provided. This infrastructure shall be compatible with the offline framework (ATHENA).

- The EH shall be scalable in the sense that increases in either the trigger rate or the event size or the required processing power for an event can be accommodated by increasing only the hardware resources. It shall be independent of the processor architecture.
- The EH shall provide the framework to recover from hardware or software failures while minimising the risk to loose events being processed.
- The EH shall continue to provide its functionality in case of a failure of the Supervision system.

The design of the EH has been made according to the following principles:

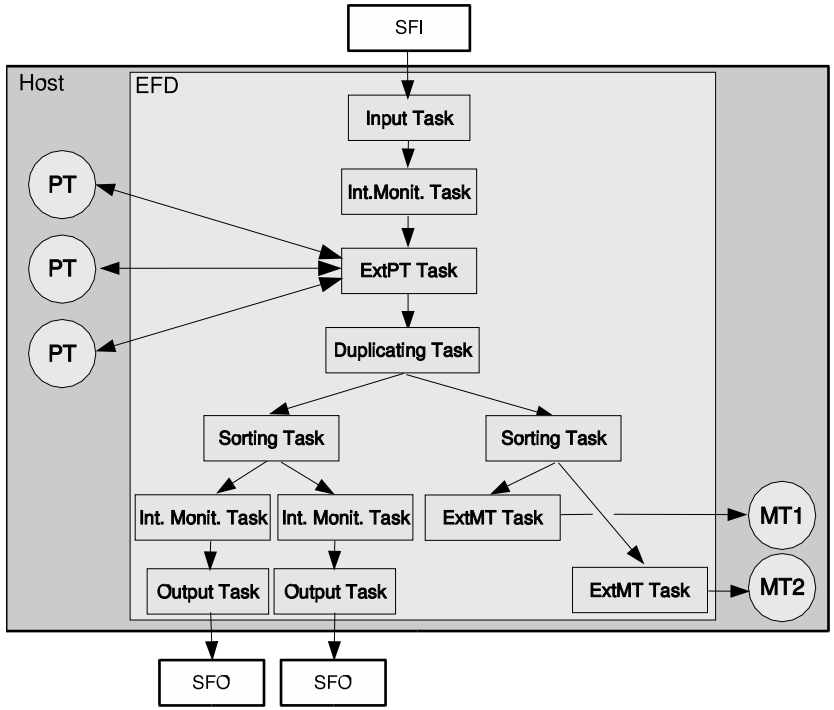
- data flow in the EH and data processing are provided by separated entities
- the flow of events is data driven, i.e. there is no data flow manager to assign the event to a specified target
- data copy on a given processing node is avoided as much as possible to save time and CPU resources

Data movement between the different phases of the processing chain is provided by the so called Event filter Dataflow process (EFD), while the processing is performed in independent Processing Tasks (PT). There is one EFD process by processing node. One or several PTs can connect to EFD at different stages of the processing chain.. Event passing is made by shared memory mapped on a local file, and synchronisation is ensured by UNIX sockets. Details can be found in [3] and [4].

9.3.2.1 Event Filter Dataflow

The processing of the events is decomposed in steps which can be configured dynamically. Every step may provide a basic functionality: event input or output, event sorting, event duplication, internal processing (e.g. for monitoring purposes), external processing, etc...

The different stages of the processing chain are implemented by "tasks". All "tasks" are derived from a base class `Task`. Possible derived classes are `InputTask`, `OutputTask`, `CounterTask` (which monitors the number of events traversing it), `ForwardingTask`, or `EndTask`. This list is not exhaustive. Some Tasks provide an interface with the external (with respect to EFD) PTs An example of an EFD implementation is given in Figure 1.

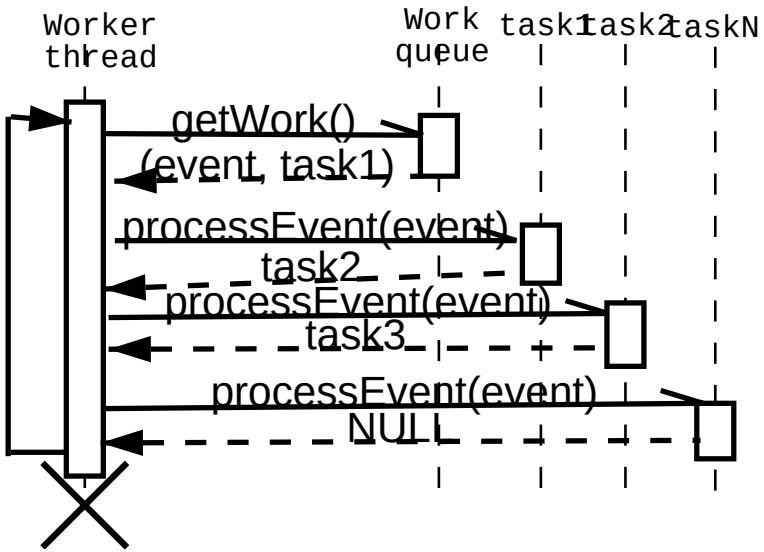


In this example, one has first an *Input Task* which makes the interface with the main Data Flow system. Events are then counted in an *Internal Monitoring Task*. The *External PT Task* provides the interface for synchronisation and communication with PTs in charge of performing the actual selection. Events which have not been tagged as rejected by the PT are then duplicated. In one of the paths, events are first counted then passed to Output Tasks to be sent to permanent storage. In the other path, on which prescaling may be applied, events are made available to different monitoring tasks according to the tag they have received during the selection in the PT.

There is no copy of data between the processing steps. Events are mapped by the `InputTask` in shared memory (`SharedHeap`) mapped on a file. Only pointers are passed to the different processing entities. The file mapping the shared memory segment ensures that data is saved by the operating system in case of problem. The information produced by the external PTs can be made available to other (monitoring) tasks if it is stored in the `SharedHeap`.

The tasks are daisy chained in the sense that each task knows the identity of the next task to execute for the current event. The `Task` base class has a method named `processEvent()` receiving a reference on an event pointer and returning a pointer on the next Task to execute. The backbone of the chaining mechanism is a `Worker` thread which first extracts an event from a Work Queue. The `getWork()` method returns from the Work Queue a pair (Event pointer, Task

pointer). It calls then the `processEvent` method of the Task passing the pointer to the Event. After processing, the method returns the pointer on the next Task, or the `NULL` pointer if it was the last task in the chain. Figure 2 shows the sequence diagram corresponding to this mechanism.



9.3.2.2 Processing Task

Processing Tasks run on every processing node as independent processes. They use the offline framework ATHENA to run the selection algorithms for the strategy described in Chapter 4.

Event passing between PT and EFD is done via the `SharedHeap` described in previous section. Synchronisation makes use of UNIX sockets. After having connected to the EFD process, the PT can request an event to the "external Task". It receives a pointer to a read-only region of `SharedHeap`. When processing is completed, PT returns an answer to the "external Task" under the form of a string. This answer is then used to decide which step will be executed then in the processing chain (event sent to permanent storage, deleted, used for monitoring purposes, etc...). PT can request a writeable block in `SharedHeap`, where it can store additional information produced during processing. If the event is to be sent to permanent storage, EFD will append this data to the raw event.

To enter more in the details, communication between EFD and PT is done via the standard ATHENA service `ByteStreamCnvSvc`. Input is made by selecting `ByteStreamEFHandlerInputSvc` (instead of `ByteStreamFileInputSvc`). In the main event loop, the `nextEvent` method gets a pointer to an event in the `SharedHeap`, casts it to the standard Event format Library (EFL) FullEventFragment and passes it to `ByteStreamCnvSvc`. Producing EF additional information makes use of the standard EFL to define an "EF sub-detector fragment" consisting

in the standard "ROS/ROB/ROD" fragment suite. One ROD fragment is dedicated to the EF result itself. Other ROD fragments may be defined to contain serialised EF PESA reconstructed objects. Finally, output is made by selecting the standard `ByteStreamEFHandlerOutputSvc`. The EF result is accessed in the transient event store and the answer is checked. If output is requested, then it gets a pointer to the EF sub-detector fragment and it serialises it to the requested `SharedHeap` extension. Finally, the EF result is passed back to EFD which determines the next step of the processing chain.

9.3.3 Supervision

9.3.3.1 Design

A detailed list of requirements can be found in [5]. This list is based on the analysis of constraints coming from other systems and on some first and general uses cases. These requirements are summarised here.

Some constraints arise from the working environment:

- •the Supervision system must work in full connection with the general ATLAS TDAQ control. In particular, it must map the finite state machine of the TDAQ Run Control. It must comply with the partitioning system.
- •the Supervision system must provide a user interface for the crew on shift. The interface must be as user friendly as possible while providing the tools for expert work during both the commissioning and steady operation phases.

The mandates of the Supervision system are:

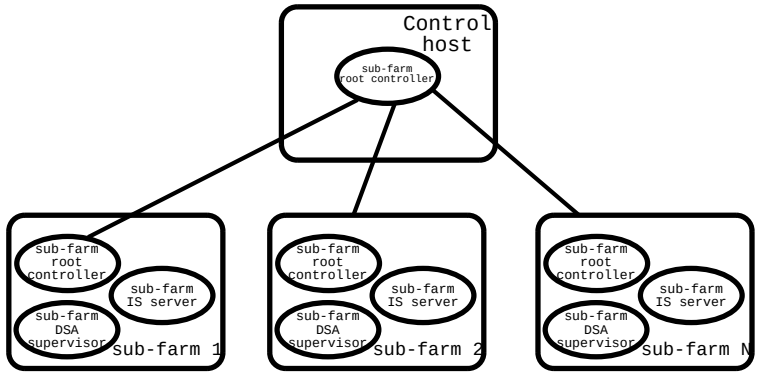
- configure the machines
- configure the software
- provide the Run Control facilities
- provide error handling and recovery facilities, as well as some bookkeeping facilities
- monitor the EF processes
- provide the user with access to the information data produced in the PTs
- possibly provide some farm management facilities (this is still an open question), i.e. hardware monitoring, operating system maintenance, code distribution on many different nodes, etc...

In addition to standard requirements on robustness and scalability, the design of the Supervision system must be flexible to cope with the evolution which will result from the better understanding of the system obtained by exercising it. In particular, the global ability of the EF system to provide a functionality good enough for allowing data taking is a concept which is bound to evolve with time when new hardware and software is used.

More details on the design and implementation of the Supervision may be found in Chapter 13. It makes use of the toolkit provided by the Online Software (see Chapter 10). A tree-like structure has been adopted. The EF Farm is organised in sub-farms, each of them is attached to a given SFI. The Run Control hierarchy consists of a root top level controller and one child controller per sub-farm. All ancillary duties related to process management are performed by a server lo-

cal to each sub-farm (the so-called *DSA Supervisor* of the Online Software suite). A local IS server allows to exchange information with other sub-systems (Figure 3).

s



Scalability and performance tests have been performed on the ASGARD cluster at ETH Zurich and on CERN-IT clusters. Results are given in [6] and [7]. They demonstrate the ability of the system to control farms the size of which reaches 1000 processors (on quad-boards).

Note: one could have here a figure showing the response time for RC transitions as a function of the number of controlled nodes e.g. figure 6 of [7]

9.3.4 Extra functionality possibly provided by EF

Although it is not strictly speaking part of the HLT, it is worth mentioning some functionality which can be provided by the EF at a rather low cost in term of resource usage. The idea is to take profit of some CPU consuming calculations which have been made for the sake of selection (mainly reconstruction) and which can be re-used for monitoring and / or calibration / alignment purposes. This could be done

- directly in EFD context (by-products of calculations performed for selection, in the filtering tasks or in independent monitoring tasks). This functionality has been illustrated in Section 9.3.2.1
- or in dedicated parts of the Farm, specially fed by the main Data Flow, and working under the control of the EF supervision

More details on monitoring in EF can be found in Chapter 7.

9.4 Event selection software

NEEDS FURTHER WORK FOR A PROPER INTRODUCTION!

The key roles of the event selection software (ESS) are ‘‘event selection’’ and ‘‘event classification’’. Abstract objects representing candidates of e.g. electrons, jets, muons and $J/\psi \rightarrow e^+e^-$ are

reconstructed from event data by a particular set of HLT algorithms and applying a set of cut parameters. An event is selected if the reconstructed objects satisfy at least one physics signature a given the trigger menu. At both stages, the LVL2 and the EF, events are rejected if they do not pass any of the selection criteria designed to meet the signal efficiency and rate reduction targets of the trigger. The boundary between LVL2 and EF is not precise from a physics event selection point on view. Indeed, flexibility in setting the boundary should be retained in order to profit from the complementary features of both trigger steps.

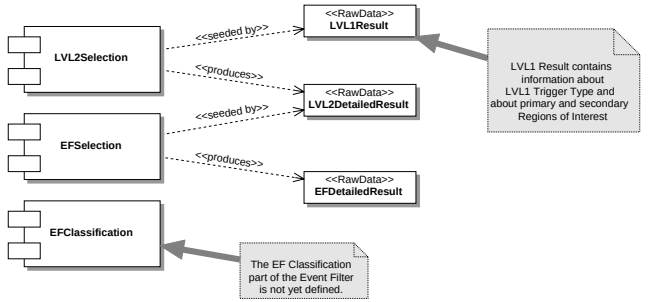


Figure 9-6 A component diagram of the high level trigger selection chain with the two steps of the LVL2 and EF selection.

The basic structure of the HLT selection chain is shown in figure??? in a simplified form. The starting point for the HLT is the LVL1 result. It contains the LVL1 trigger type and the information about primary ROIs that caused the LVL1 accept, plus secondary ROIs not considered for the LVL1 accept. Both types of ROIs are used to seed the LVL2 selection. The concept of seeded reconstruction is fundamental to the LVL2, apart from the special case of B-physics.

The LVL2 result plays a similar role for the EF as does the LVL1 result for the LVL2. The LVL2 result provides the means to seed the EF selection. It should also be possible to seed the EF directly with the LVL1 result in order to study for example the LVL2 performance. The EF and the LVL2 results will be appended to the raw event data.

A yet to be further defined component is the EF classification. It may include special selections for calibration events and for new physics signatures, i.e. a discovery stream. The LVL2 and EF results include those physics signatures from the trigger menu which were satisfied and higher level reconstruction objects. The EF result can be used to assign tags to the events or even assign them to particular output streams.

The event selection software involves both the infrastructure or framework and the selection algorithms. The latter are to be provided either by the PESA group or, in case of the algorithms for the EF, by the offline reconstruction group. Major parts of the online reconstruction will have to be based on offline reconstruction algorithms. This is an important constraint for the design of the Event Selection Software.

In the online the Event Selection Software will run in the software environments provided by the LVL2 processing unit and the EF processing task, as is shown in figure???. Therefore the event selection software needs to comply with the online requirements, like thread safety, online system requirements and services, as well as online performance goals.

It is essential though that the event selection software is also able to run directly in the offline environment ATHENA (ref.???) to facilitate development of algorithms, to study the boundary between LVL2 and EF and to allow performance studies for physics analysis. Therefore the event selection software needs to comply with the control framework and services that are provided by the offline software architecture team. For this reason the ATHENA framework was chosen as the framework to implement the event selection software inside the EF processing task and in the modified form of the PESA steering controller in LVL2 processing unit.

MAY ADD A SUBSECTION ON THE REUSE OF OFFLINE SOFTWARE IN THE TRIGGER, SHOULD DEFINITELY BE BETTER WORKED INTO THE TEXT.

9.4.1 Package Dependencies in the Online System

HERE ONE HAS TO REWORK THE DIAGRAMS TO TAKE ATHENA AS A FRAMEWORK INTO ACCOUNT. QUESTIONS ARE WHERE TO PUT THE BS CONVERSION SERVICE, STORE-GATE ETC.... SEE WITH FRED, SAUL..

In figure??? the package diagram is shown for the event selection software running in the LVL2 processing unit. The PESA steering controller uses the interface of the event selection software to request the LVL2 selection on the next LVL1 result. The event selection software needs to access ROB data fragments and uses meta data in order to obtain the LVL2 decision. The ROB data requests are sent via an interface of the ROB data collector. Services are used to access meta data. These include the geometry, conditions and B-Field map information needed for the algorithmic trigger processing of the event. Monitoring services serve purposes like histogramming and messaging.

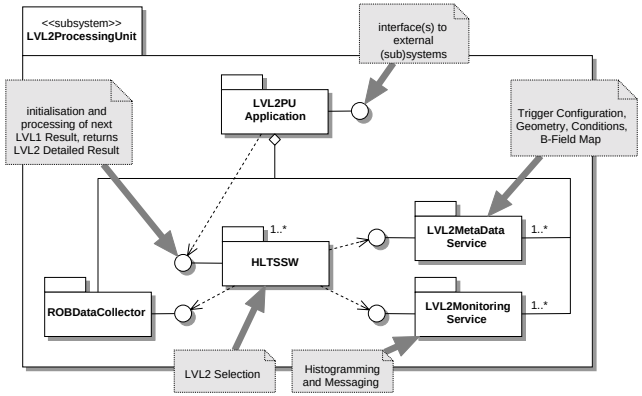


Figure 9-7 Package diagram showing the dependencies of the event selection software performing the LVL2 selection in the LVL2 processing unit. *DIAGRAM NEEDS UPDATE (HLTSSW-ESS, PSC,...)*

QUESTION: WE DESCRIBE THE PSC AS AN ENHANCED "VERSION" OF ATHENA? WHERE DO WE BREAK UP THE LONDON SCHEME?

No external dependencies are shown in the figure. The communication with external systems and subsystems, including the LVL2 Supervisor and the ROS, are hidden from the event selection software by means of the PESA steering controller. The event selection software can be initialised via its interface at the start of a "RUN". Likewise, the PESA steering controller requests the event selection software to process a given LVL1 result. Note that to optimize the use of available computing resources several LVL1 results are processed in parallel in different threads.

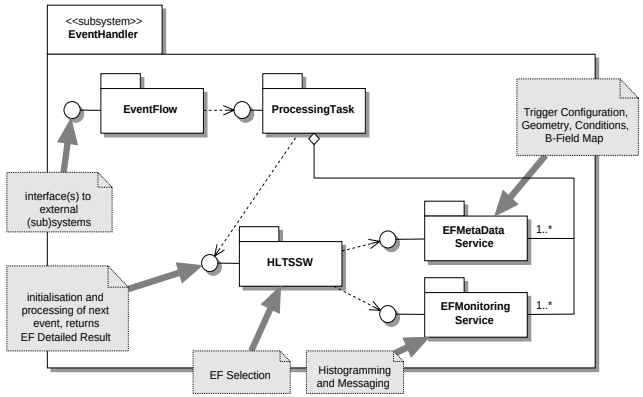


Figure 9-8 Package diagram showing the dependencies of the event selection performing the EF selection in the Event Handler. *DIAGRAM NEEDS UPDATE (HLTSSW-ESS, Data-Flow,...)*

In figure??? the corresponding package diagram is shown for the event selection software running in the event handler. The data flow has an interface to the external subsystems for the communication with the event filter IO. The data flow package transmits the event to the processing task, which requests the EF selection for the event using the interface of the event selection software. No equivalent of the ROB data collector is needed by the processing task, because it is carried out after the event building. Again meta data and monitoring services are needed for the algorithmic trigger processing. Again, the dependencies on external systems and subsystems are hidden from the event selection software, including event filter IO and run control. It is foreseen to run a single EF selection per processing task.

In both subsystems the event selection software depends on interfaces of the meta data and monitoring services. The use of ATHENA as a common framework allows for the same or similar interface definitions running offline or online

9.4.2 Package Dependencies in the Offline

The package diagram for the event selection software running offline in ATHENA is shown in figure???. In the offline the task of the HLT is to emulate the full online selection chain. This requires also to use the emulation of LVL1 \cite{LVL1}, which is provided by the LVL1 system. Hence three so called top level ATHENA algorithms are needed, namely the LVL1 trigger emulation and two instances of the event selection software. The LVL1 trigger emulation provides the LVL1 result. The first instance of the event selection software is configured to execute the LVL2 seeded by the LVL1 result, the second to execute the EF selection seeded by the LVL2 result.

NEEDS MAJOR UPDATE! ATHENA IS THE MAIN FRAMEWORK, ROB DATA PROVIDER AND BS CONVERSION SERVICE INSIDER HLTSSW. MORE TEXT IS NEEDED HERE!!! WHAT IS THE OFFLINE DATA FLOW VIA BS CONVERSION SERVICES, BS FILES...

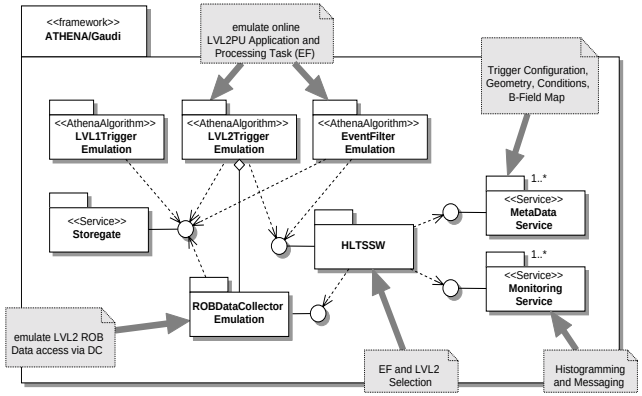


Figure 9-9 {Package diagram showing the dependencies of the event selection software running the LVL2 and EF Selection in the offline. *DIAGRAM NEEDS UPDATE! ATHENA IS THE MAIN FRAMEWORK, ROB DATA PROVIDER AND BS CONVERSION SERVICE INSIDE HLTSSW...*

9.4.3 An Overview of the Event Selection Software

NEEDS UPDATE TO ACCOUNT FOR THE USE OF ATHENA ONLINE/OFFLINE, BS SERVICES...

In figure??? the package diagram of the event selection software is shown. The sub-packages are:

- The **Steering** controls the selection software. It ‘‘arranges’’ the algorithm processing for the event analysis in the correct order, so that the required data is produced and the trigger decision is obtained. The steering implements the interface to the PESA steering controller (when running in the LVL2 processing unit) and to the processing task (when running in the event handler). The same interface is used when running in the offline framework ATHENA.
- The event data is structured following the **Event Data Model** (EDM). The EDM covers all data entities in the event and their relationships with each other. The data entities span from the raw data in byte stream format (originating from the detector RODs), the LVL1 result and all other reconstruction entities up to the LVL2 and EF result. The EDM is therefore the ‘‘language spoken’’ by the software to communicate information about the event.
- The **HLT Algorithms** are used by the steering to process the event and to obtain the data on the basis of which the trigger decision is taken. The dependency on offline algorithms is especially important for the implementation of the EF. Trigger versions of offline algorithms are derived that are adopted to online requirements. Hence, a common definition

of the EDM for the online and offline needed in order to facilitate the reuse of Offline Algorithms.

- The **Data Manager** handles all event data, including the raw data access, during the trigger processing. It therefore provides the necessary infrastructure for the EDM. For the case of LVL2 the communication with the ROB data collector to access the ROB data fragments is part of the data manager.

In summary, the EDM covers all event data entities and is used by the other parts of the event selection software, by the Steering, the HLT algorithms and the data manager, to communicate information about the event. The HLT algorithms build up the event tree in the process of the reconstruction. The result is analysed by the steering to obtain the trigger decision. The data manager has to provide an important functionality, as it supports the EDM. It provides the means of accessing the event data and for building it up. The data access patterns reflect the needs of the HLT algorithms and of the steering. Here, raw data access for example by ‘‘region’’ is clearly a requirement. In the following sections more detail is given on the event selection software sub-packages.

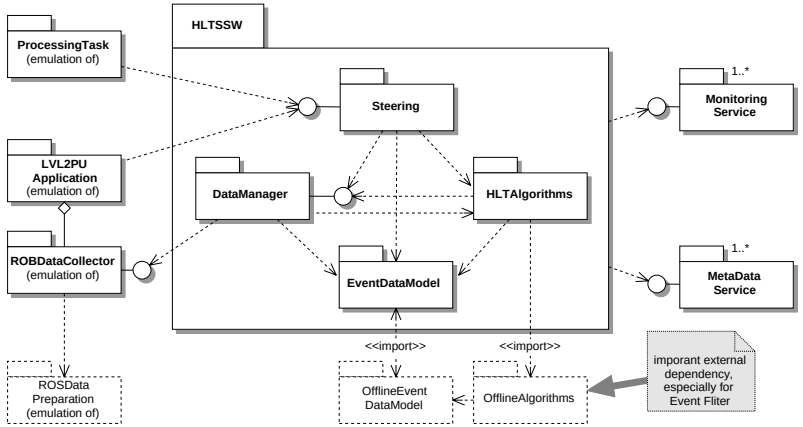


Figure 9-10 A package diagram of the event selection software, showing the dependencies of the sub-packages. The external packages are named following the online case and include their respective offline emulations. The dependencies on the offline event data model and on the offline algorithms are explained in the text. *DIAGRAM NEEDS UPDATE, EMULATIONS, DATA MANAGER, PROCESSING TASK-ATHENA EVENT LOOP MANAGER, LV2PU-PSC...*

9.4.4 The Event Data Model Sub-package

... UNIFY WITH SECTION 13, TALK WITH STEVE AND VALERIO... INTRODUCED RECONSTRUCTION INPUT OBJECTS.

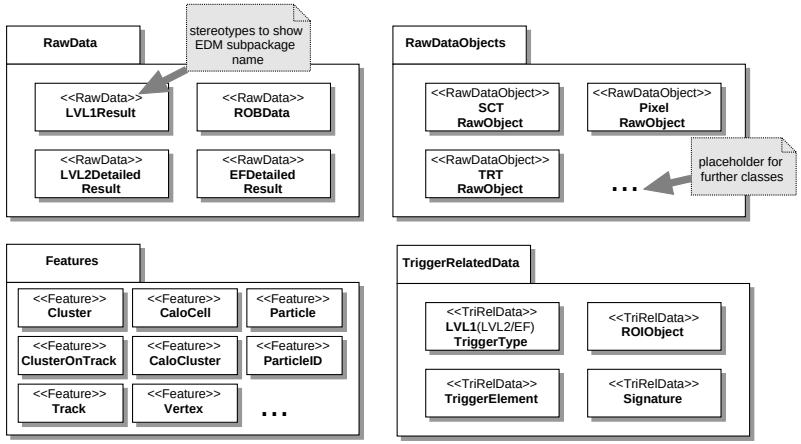


Figure 9-11 EDM package diagrams showing the Raw Data, Raw Data Objects, Features and Trigger Related Data sub-packages. In all cases the stereotypes in the class names are used to indicate the package they belong to. *NEEDS UPDATE FOR RecInputObjects*

The HLT is a two stage software trigger that selects events by means of reconstruction. It is guided by the ROI information provided by the LVL1 system. All event data classes and their relations are summarised in the EDM. The EDM of the event selection software is closely coupled to the offline EDM, especially because offline algorithms are the basis of the EF selection. The EDM is therefore being developed in close contact with the offline EDM, detector and reconstruction groups. The EDM classes are grouped into 5 sub-packages:

- **Raw Data** coming from the ROS and the LVL1 system is in byte stream format. The LVL1 result, the LVL2 and EF results, as well as ROB data from the sub-detectors and from the LVL1 system are raw data. Part of the raw data formats are headers and trailer from the RODs, ROBs, ROSs and DC system. Note that for a given sub-detector several raw data formats might be used, e.g. different formats for depending on the LVL1 trigger type. This includes different the data content or the compression schemes.
- The **Raw Data Objects** are an object representation of the raw data from the different sub-detectors. In the offline the raw data objects (e.g. SCT raw objects) are created at input to the reconstruction chain. The object creation poses an overhead, especially for LVL2. Therefore in the current design the RDOs are only used in the EF. The LVL2 converts the raw data directly to low level features, i.e., calorimeter cells or clusters. As it is the case for the raw data, raw data objects are highly sub-detector dependent information.
- **Features** are all type of reconstruction data derived from the raw data objects or from other features with increasing levels of abstraction. This includes all offline reconstruction EDM classes. They range from data produced after calibration and clustering up to reconstructed quantities such as tracks, vertices, electrons or jets.
- **MC Truth** information. Together with the first three sub-packages of the EDM, the MC truth is common to both the event selection software and the offline reconstruction. It is needed primarily for debugging and performance studies.

- Another part of the EDM is **Trigger Related Data**. It comprises ROI objects and the LVL1(LVL2/EF) trigger type (???ref{LVL1Result}), as well as so called trigger elements (TEs) and signatures. A TE labels a set of reconstructed features and implies (by its label) a physical interpretation of these features, such as a particle, missing energy or a jet. It represents thus an even higher level of abstraction. TEs are the objects used by the steering to guide the processing and to extract the trigger decision.

A package diagram for raw data, raw data object, features and trigger related data is given in figure???. The stereotypes in the class names are used to indicate the package name. This convention will also be used in following diagrams.

MORE EDM IMPLEMENTATION DETAILS ON THE FEATURES/RDOs ARE TO BE PROVIDED BY SECTION 13 TOGETHER WITH DESCRIPTION OF THE BS CONVERTER AND THE ALGORITHM IMPLEMENTATION.

9.4.4.1 Object Relations and Event Structures

During the trigger processing the event is built up. Each event contains instances of EDM classes. An important aspect of the EDM are the relations between the different data classes. Navigability of such relation is essential for the algorithm processing, because navigation is used to analyse and built upon the previously reconstructed event fragments. Role names of the class relations are used to indicate the nature of the association.

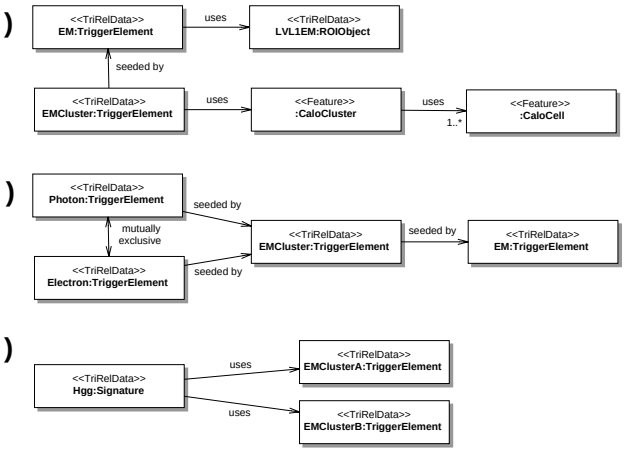


Figure 9-12 Examples of relations between instances of EDM classes, which will be typical for an event processed by LVL2.

In figure??? typical examples of instances of EDM classes and their relations are given for an event processed by LVL2. Figure???.a shows a TE that has a ‘‘uses’’ relation to a LVL1 ROI. Another TE, this one having a uses relation to a whole tree of reconstructed features, is ‘‘seeded by’’ the first one. In figure???.b an example for a ‘‘mutually exclusive’’ relation between TEs is shown. Finally in???.c a valid signature uses two TEs. The full event tree is constructed during the LVL2 or EF trigger processing out of fragments like the ones displayed here. The analysis of

this tree provides means for seeding the trigger reconstruction and to derive the trigger decision.

9.4.5 The HLT Algorithms Sub-package

... UNIFY WITH SECTION 13, TALK WITH STEVE AND VALERIA...ADD REFERENCE TO CHAPTER 13, GET LANGUAGE UNIFIED..

The EDM is tightly coupled to the HLT algorithm processing. The EDM definition helps to support a modular algorithm design. Compared to the offline situation, where this statement also holds, it is even more important for the event selection software, because the trigger selection is data driven

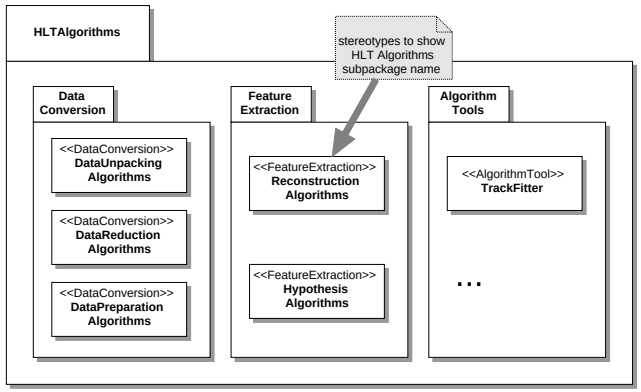


Figure 9-13 A package diagram of the HLT algorithm sub-package. *DIAGRAM NEEDS TO BE UPDATED. DATA PREPARATION SHOULD BE STRUCTURED INTO CONVERTERS...LVL2.*

Figure??? shows three HLT Algorithm sub-packages. They provide a rough structuring of the trigger reconstruction. Again, stereotypes are used in the figure to indicate the corresponding sub-package. The three sub-packages are:

- **Data Conversion** comprises algorithmic code for raw data conversion into objects that are input to reconstruction. The task of this type of tools involves sub-detector specific information. Hence, sub-detector groups are responsible for the implementation and maintenance of the code, taking the HLT requirements into account. Data conversion ends with ‘‘low level’’ features, which are input to the second category of HLT algorithms.

  In the current design the organisation of the data conversion is different for the LVL2 and the EF. The EF follows closely the offline reconstruction chain that starts from raw data objects. Hence the raw data gets first converted into object form and is then processed in order to obtain clusters or calorimeter cells. In the LVL2 the step of object creation is omitted in order to avoid the overhead. Here the data preparation is going in one step from raw data to ‘‘low level’’ features.

  UPDATE USING THE NAMES AFTER CHANGING THE FIGURE.

  The boundary between Data Conversion and Feature Extraction is not exact, but note that only Data Conversion will see the Raw Data or Raw Data Objects.

- **Feature Extraction** algorithms operate on abstract features and trigger related data to refine the event information. They built upon the data conversion output and extract the necessary input data to derive the trigger decision. Two types of algorithms are distinguished in the feature extraction sub-package. **Reconstruction Algorithms** process features and produce new types of features, just like offline reconstruction algorithms. Trigger specific is the use of the information in ROI objects to restrict the processing to geometrical regions of the detector, which were identified by the LVL1 system. The TEs used by the steering to ‘‘seed’’ the reconstruction algorithms represent these trigger relevant aspects of the event. The seeding mechanism is discussed in the next subsection. The second type of algorithms are **Hypothesis Algorithms**. Their task is similar to particle identification. A hypothesis algorithm validates the physics interpretation implied by the label of the TE based on the reconstructed features. An example is the validation of an ‘‘electron’’ using a reconstructed calorimeter cluster and a track.

- It is beneficial to structure the algorithm processing in such a way that algorithms from a library of **Algorithm Tools** carry out common tasks such as track fitting or vertex finding.

There is an important difference between the offline and the HLT reconstruction, especially for LVL2. The ROI based approach implies a data driven event reconstruction. In contrast for the offline, any HLT algorithm in a reconstruction sequence may be executed several times per event, once for each ROI. Therefore a modular structure of the offline algorithms is necessary.

Certain types of data manipulation are only possible at a later stage in the reconstruction, even though they are in principle part of the detector specific processing. These operations violate in a certain sense the three category structure given above. An example is the transformation of 1 or 2 dimensional objects, such as TRT drift circles or silicon clusters, into the ATLAS global frame by means of alignment and conditions constants from the Conditions Service. This is possible only at the time of track fitting, because the exact position can be deduced only on the basis of external predictions that use the track itself. Operations with this characteristics should be identified and standard methods developed to separate those detector specific operations from the feature extraction algorithms.

9.4.5.1 The Seeding Mechanism

The HLT algorithms are the building blocks of the reconstruction that provides the data to derive the trigger decision. Logically the trigger processing is done starting from a LVL1 ROI using predefined sequences of algorithms. A so called ‘‘seeding’’ mechanism is needed in order to guide the reconstruction to the event fragments relevant for preparing the trigger decision.

It is beneficial not to couple directly the steering of the trigger selection to the details of the feature extraction algorithms. No ‘‘micro’’ sequencing should be done on the level of features, e.g. single calorimeter cells or silicon clusters. These complex event details are better handled by the algorithms themselves. Therefore TE are to be used to characterise with their label the abstract physics objects, e.g. ‘‘electrons’’ or ‘‘jets’’. Besides this label a TE does not have any properties or states of its own, because the list of possible states or properties is not well defined a priory. In-

stead it is the ‘‘uses’’ relation by which the TE is associated to the concrete event data, that should provide all necessary information to the algorithms.

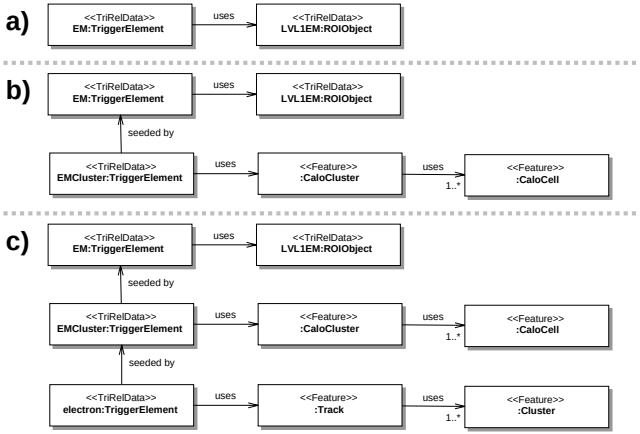


Figure 9-14 Three diagrams showing fragments of the event fragment associated to one ROI at different stages of trigger processing. See text for details.

Figure??? three diagrams are shown of the event fragment associated to one ROI at different stages of trigger processing. The evolution of the fragment from step to step illustrates how the seeding mechanism works. The upper part???.a shows an example of an electro-magnetic ROI Object from the LVL1 calorimeter trigger. The ROI object is ‘‘used’’ by an ‘‘EM’’ TE to label it for the HLT trigger processing. Starting from this input TE it is he task of the first HLT reconstruction step to validate the physics hypothesis of having an electromagnetic cluster above a certain threshold. For this hypothesis an output TE is created with the corresponding label by the steering. The output TE is linked to the input TE by an ‘‘seeded by’’ relation. The steering then executes the HLT algorithm, giving it the output TE as an argument. It is the task of the HLT algorithm to validate this output TE.

A sequence diagram for the cluster finding algorithm is shown in figure???. The algorithm gets as a seed the output TE that provides the means to direct the pattern recognition to the region of interest. The first thing the algorithm needs to do is to obtain information about the geometrical position of the LVL1 ROI. It can do so by navigating via the ‘‘seeded by’’ and ‘‘uses’’ relation from the output Te to the input TE to the ‘‘LVL1EM’’ ROI object. The algorithm obtains η and ϕ from the ROI and does its pattern recognition work. In our example it creates a calorimeter cluster from a set of calorimeter cell objects. These objects are linked the output TE to record the new event information associated with this ROI for later processing. Based on the reconstructed cluster the algorithm decides it all cuts are passed to validate the hypothesis. The algorithm needs to transmit the result to the steering, which is done by ‘‘activating’’ the output TE in case of a positive decision. The steering will thus ignore all in-active TEs for further processing. The event fragment at the end of the algorithm execution is shown in figure???.b

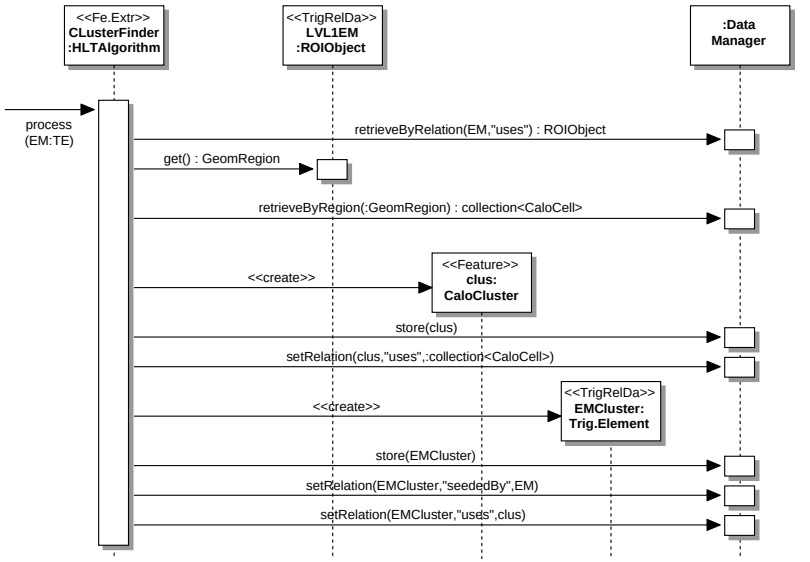


Figure 9-15 Sequence diagram for the cluster finding feature extraction algorithm seeded by an ‘‘EM’’ TE. *DIAGRAM NEEDS UPDATE FOR THE NEW SCHEME OF ACTIVATION OF THE OUTPUT TE, HENCE NAVIGATE BACK TO INPUT TE AND ROI, GET ETA/PHI, RECONSTRUCTION, SAVE CLUSTER AND TRACKS, ACTIVATE TE IF CUT PASSED. - MONIKAS PLOT*

The next algorithm in the example is a electron track finding algorithm. It is seeded with a new output TE with the label ‘‘electron’’. The algorithm is able to navigate the full tree of data objects, as shown in figure???.b, to access the necessary information. To validate the ‘‘electron’’ physics hypothesis it could use the precise position of the calorimeter cluster to reconstruct the track from a set of SCT clusters. Because a track is found the ‘‘electron’’ hypothesis is validated and a TE get activated. The resulting event fragment is shown in figure???.c.

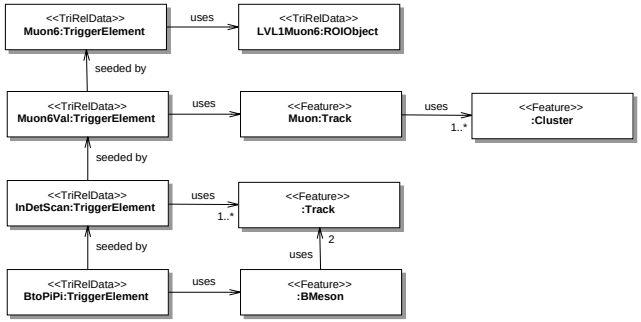


Figure 9-16 A diagram showing a fragment of the event for the case of LVL2 B-Physics trigger reconstruction. See text for details.

In figure??? a similar diagram to???.c is shown for the case of a LVL2 B-physics trigger. Here, the new aspect is the inner detector full scan. This means reconstructing all tracks in the event after the initial validation of a ''muon''. A TE ''uses'' the collection of tracks and seeds, for example, a algorithm that reconstructs an exclusive B-decay into two pions. The result of this would be a TE called ''B to PiPi''. This TE uses a B-meson and is ''seeded by'' the ''Inner Detector Scan'' TE, as shown in the figure.

9.4.6 The Steering Sub-package

... GIANLUCA TO UPDATE THE SECTION TO GET IT IN LINE FOR THE ACTUAL IMPLEMENTATION, SOME EDITING IS DONE ALREADY...

The **Steering** controls the HLT selection. It ''arranges'' the HLT algorithm processing for the event analysis in the correct order, so that the required data is produced and the trigger decision is obtained. The steering controller is the component of the Steering that provides the interface to the PESA steering controller for running in the LVL2 processing unit and to the processing task for running in the event handler. In the offline the steering is the top level algorithm executed directly by the ATHENA event loop manager.

The LVL2 and the EF selection is data driven, in that it builds on the result of the preceding trigger level. It is the task of the steering to guide the HLT algorithm processing to the physics relevant aspects of the event, using the seeding mechanism discussed in the previous section. Seen from the side of the steering, the reconstruction of an event in the trigger is a process of refining TEs, as was shown in figures ??? and???. For each TE at input several HLT algorithms were executed that validated output TEs.

The goal of the HLT selection is to trigger the interesting physics events. The selection demands that the event matches at least one so-called physics ''signature''. Each signature is a combination of abstract physical objects like ''electrons'', ''muons'', ''jets'' or ''missing energy''. It is usually requested that, for example, an electron has a minimal energy and is isolated from a jet. Translated into the event selection software a signature is nothing else but a combination of required TEs in an event. A possible syntax for writing a Signature of requested TEs is using the label ''2 x e20i'', which would mean two ''20 GeV isolated electrons''. In table??? and??? the final LVL2 and EF trigger menu is shown for the case of??? luminosity. These menus contain a complete list of such ''signatures''.

As shown in the tables, the Signatures used to select the event can be quite explicit, like 4 Jets or a Z –> ee. For each signature requiring more than one TE it is beneficial not to completely validate the first, because the second may be e.g. due to noise and therefore may fail right away at the beginning of reconstruction. The HLT algorithm processing is therefore done in steps. At each step a part of the reconstruction chain of HLT algorithms is carried out for each TE, if possible, starting with the HLT algorithm giving the biggest rejection. At the end of each step a decision should be taken, whether the event can still possibly satisfy one of the final physics signatures.

9.4.6.1 The Trigger Configuration

... THOMAS TO UPDATE FOR THE RECURSIVE HLT TRIGGER CONFIGURATION BASED ON XML FILES...

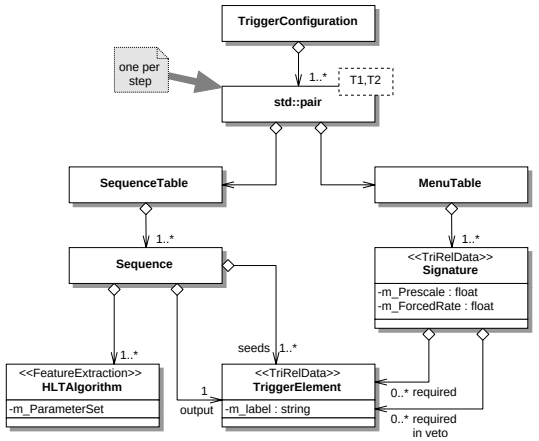


Figure 9-17 A class diagram for the trigger configuration, showing for each step a pair of sequence table and menu table.

Figure??? shows the class diagram for the **Trigger Configuration**}. It contains a **Sequence Table** and a **Menu Table** for each of the trigger steps described above. Both of these are discussed below.

The sequence table consists of a collection of **Sequences**, where a Sequence is defined as a transformation of one or more TEs, via a set of HLT algorithms, into a new TE. The steering obtains information about the specific HLT algorithm to execute for a given seed from the sequence. Each HLT algorithm in a sequence is configured with the relevant parameter set.

The menu table contains the collection of **Signatures** that could be validated in a given step. Each signature usually contains collections of required TEs and of TEs required in veto, a prescaling factor, and a forced accept rate. Some exceptions to this do occur; for example random triggers do not specify a list of requested TEs, only a forced accept rate.

It is important to provide a consistent trigger configuration at all trigger levels (LVL1, LVL2, EF). As sequences provide input and output information in terms of TEs, it is possible to derive from the final EF menu table the trigger configuration for all trigger levels. The complete list of possible sequences is sufficient to calculate the sequence and menu tables for all earlier steps. In the current implementation the recursive generation of a consistent trigger configuration is done for the EF and LVL2 based on XML sequence and menu tables. In the future the LVL1 configuration will be derived from the set of input TEs for the first LVL2 step.

9.4.6.2 An Overview of the Steering

... GIANLUCA TO UPDATE THE SECTION, THE NAMES AND CONVENTIONS ACCORDING TO THE CURRENT IMPLEMENTATION. THE DESCRIPTION OF THE INTERFACE NEEDS CORRECTION...

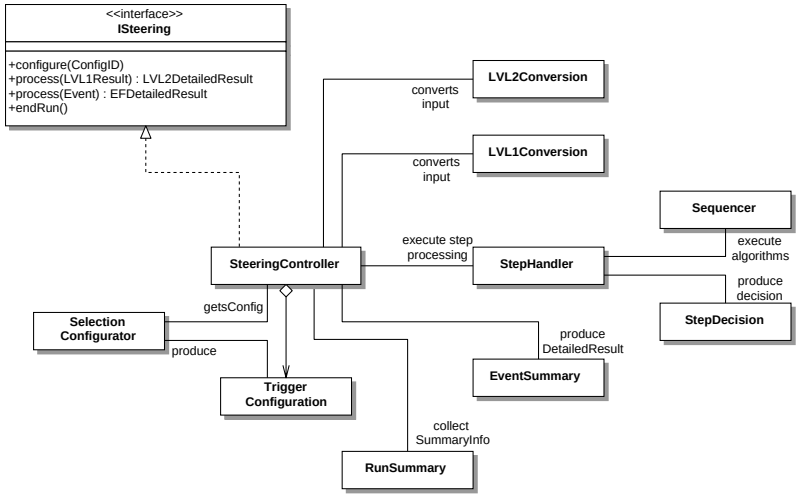


Figure 9-18 A class diagram for the Steering, including the interface to wards the LVL2PU and the Processing Task. *DIAGRAM NEEDS UPDATE, GIANLUCA TO FOLLOW UP.*

In figure??? a class diagram for the steering of the event selection software is given. Its interface to the PESA steering controller and the processing task provides the necessary methods to configure the event selection software at the beginning of a ''RUN'', to execute the LVL2 or EF selection, and to end a ''RUN''. The interface is implemented by the **Steering Controller**.

The task of the **Trigger Configurator** is to provide the trigger configuration. This task is carried out during the initialisation phase of the trigger system (before the start of a ''RUN''), which is especially important for LVL2.

In the LVL2 the steering controller uses the **LVL1 Conversion** to convert the LVL1 result (i.e. Raw Data) into ROI objects and prepares the selection by creating the TEs needed for the seeding mechanism. In case of the EF the corresponding LVL2 **Conversion** translates the LVL2 result to prepare the EF selection. The trigger processing in steps is then carried out by the **Step Handler**. It uses the **Sequencer** to execute the HLT algorithm in the sequences. The step handler executes the **Step Decision** to compare the result of the algorithmic processing, given in terms of TEs, with the signatures to decide whether or not to reject the event. The next step is processed by the step handler only if the event is still accepted, until all steps are executed. The role of the **Event Summary** is to produce the LVL2 or EF results, depending on where the event selection software is running.

The steering controller uses the **Run Summary** at the end of a ''RUN'' to collect summary information for monitoring purposes.

9.4.6.3 Configuration of the event selection software

... THOMAS TO UPDATE THIS SUBSECTION...

In figure??? a sequence diagram is shown for the selection configuration. Its task is to provide a trigger configuration, which is then used by the step handler to carry out the trigger selection. The input to the selection configuration are XML menu and sequence tables. Using this information the EF and LVL2 trigger configuration is created recursively.

PUT THIS AT THE RIGHT PLACE: The sequences in the sequence tables contain HLT algorithms. They need to be created and configured using a parameter set that is part of the configuration data. It is part of the HLT algorithm initialisation to access meta data via dedicated services. For LVL2 it is essential that also the access to meta data is limited to this phase.

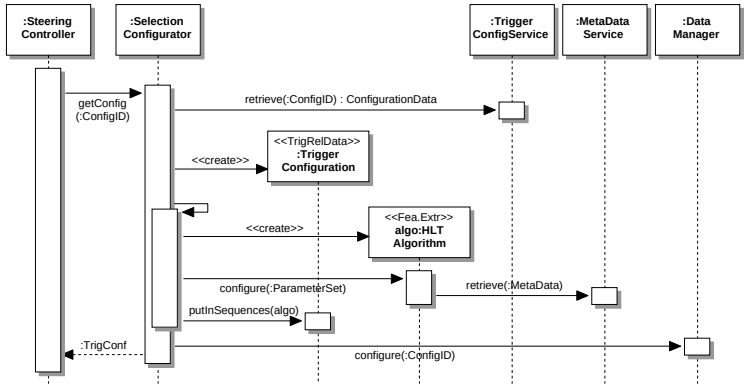


Figure 9-19 A sequence diagram for the Selection Configurator that provides the Trigger Configuration. *THOMAS TO UPDATE THIS DIAGRAM.*

9.4.6.4 The LVL1 Conversion

... GIANLUCA TO UPDATE THE SECTION...

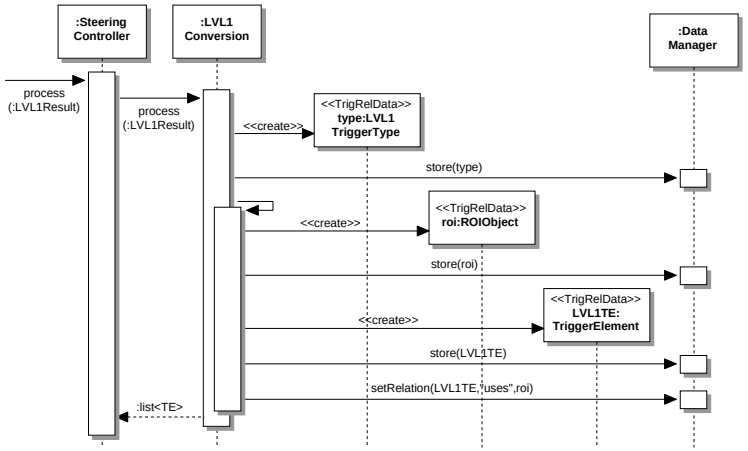


Figure 9-20 A sequence diagram for the LVL1 conversion. *GIANLUCA TO UPDATE THE DIAGRAM*

In figure??? a sequence diagram is shown for the LVL1 conversion. It is used by the steering when running in LVL2. In order to prepare the LVL2 selection the LVL1 result \cite{LVL1Result} is converted into the LVL1 trigger type and into ROI objects. Finally, a TE is create for each ROI Object ''using'' it, as has been discussed in figure???.a. The LVL1 Conversion provides this list of TEs to the steering as the starting point for the step processing.

The LVL1 conversion has to know the configuration of the LVL1 system to translate the raw data. Therefore the LVL1 conversion is to be developed in close contact with the LVL1 system. In case of running the EF selection it is the task of the corresponding LVL2 conversion to decode the LVL2 result to enable LVL2 seeding of the EF.

9.4.6.5 The Step Processing

MONIKA AND GIANLUCA TO UPDATE THIS SECTION.

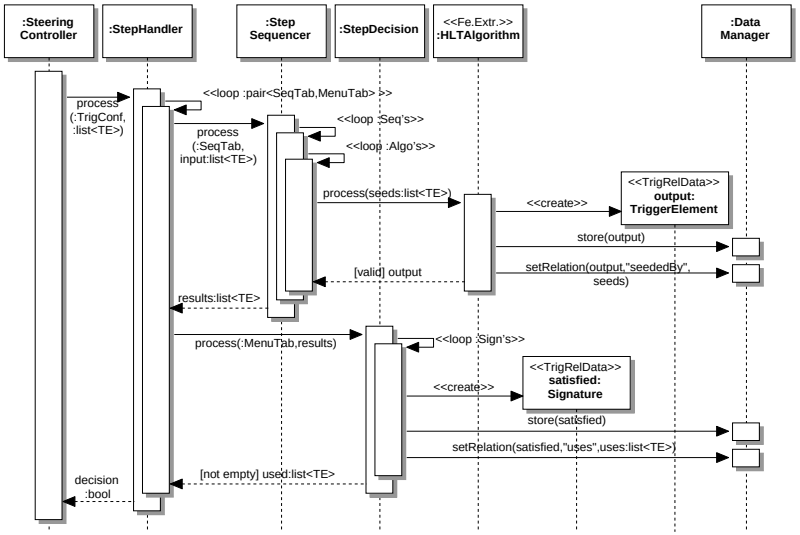


Figure 9-21 A sequence diagram for the step processing showing the role of the step handler, of the sequencer and of the step decision. *UPDATE DIAGRAM FOR THE TE ACTIVATION SCHEME -MONIKA.*

In figure??? a sequence diagram is shown for the step handler carrying out the step processing for an event. The step handler gets as arguments the trigger configuration and the initial list of TEs produced by the LVL1 (LVL2) conversion. Each step corresponds to a pair of a menu table and a sequence table. The reconstruction part for each step is controlled by the sequencer. It executes HLT algorithms in sequences to further process the event and to refine the content in terms of TEs. The step decision decides based on the outcome of each reconstruction step whether or not to reject the event.

Depending on the input list of TEs at the beginning of each step the sequencer executes HLT algorithms in sequences, which are defined in the sequence table. The task is to ''seed'' the HLT algorithms in the sequences executed for all (one at a time) matching combinations of TEs out of the original list of input TEs and to request validation of the resulting output TEs. The resulting list of TEs is passed to the step decision after all sequences in the sequence table of this step have been processed. The step decision then compares the list of output TEs to the signatures in the menu table for this step. For each matching combination of TEs a signature is stored in the data manager. The signature ''uses'' these TEs in order to prepare the LVL2 (EF) result. The step decision returns a list of TEs, which have been used to satisfy at least one signature. The remaining TEs are discarded from further processing. The step handler continues processing the next step either until it is rejected by the step decision, because no signature is satisfied, or until all steps are done, in which case the event is accepted.

9.4.6.6 Obtaining the LVL2 and EF Results

... GIANLUCA TO CHANGE THE SUBSECTION, SAY SOMETHING ON THE LVL2 RESULT TRANSFER TO PSC...

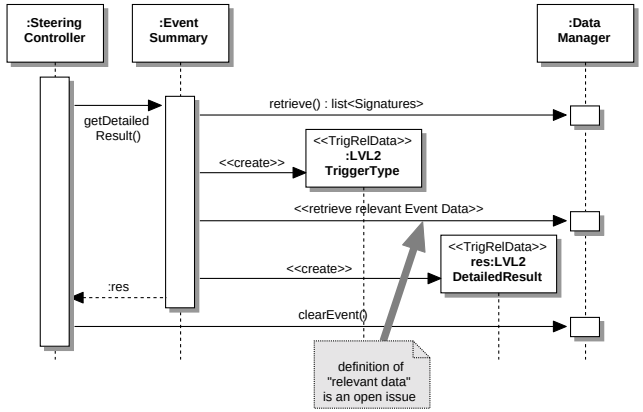


Figure 9-22 A sequence diagram for the Event Summary executed at the end of the processing of each accepted event. *GIANLUCA TO FIX THIS DIAGRAM*

The event summary is executed after the step handler for each accepted event. Its task is to produce the LVL2 or respectively the EF result based on the event information produced. As shown in figure???, the event summary retrieves the list of validated signatures from the data manager. From this the LVL2 (EF) trigger type is determined. The exact content of the LVL2 or EF results, which is to be constructed from the event information, is not yet defined.

At the end of the event analysis the data manager has to clean the event data from its memory.

... CRISTOBAL MAY PUT SOMETHING ABOUT THE NEW MONITORING STUFF HERE...

9.4.6.7 Ending a Run of the Event Selection Software

NEEDS UPDATE!!!

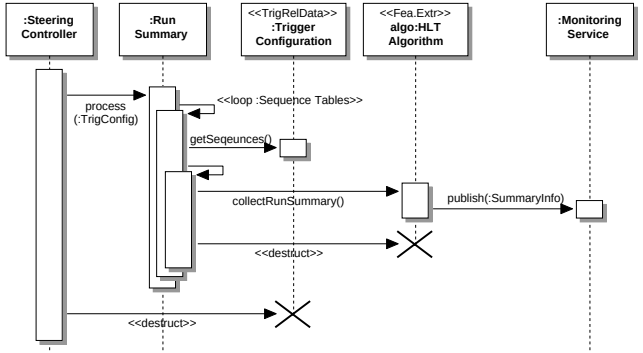


Figure 9-23 A sequence diagram for the Run Summary, which is executed at the end of a ‘‘RUN’’. *CRISTOBAL MAY UPDATE THE DIAGRAM*

The final component of the steering sub-package is the run summary. Its task is to collect the summary information from the HLT algorithms, which are in the sequences of the sequence tables in the trigger configuration. This summary information is published via the monitoring services.

GIANLUCA, PLEASE CORRECT AND PUT THIS SOMEWHERE...

The run summary also destructs the instances of the HLT algorithms. Afterwards, the steering controller destructs the old trigger configuration.

9.4.7 The Data Manager Sub-package

THIS SECTION NEEDS MAJOR REWORK AS THE NATURE OF THE DATA MANAGER HAS CHANGE WITH THE USE OF ATHENA/STOREGATE. THE LONDON SCHEME AND THE BYTE STREAM CONVERSION SERVICE NEEDS TO BE INTRODUCED AS WELL AS THE REGION SELECTOR.

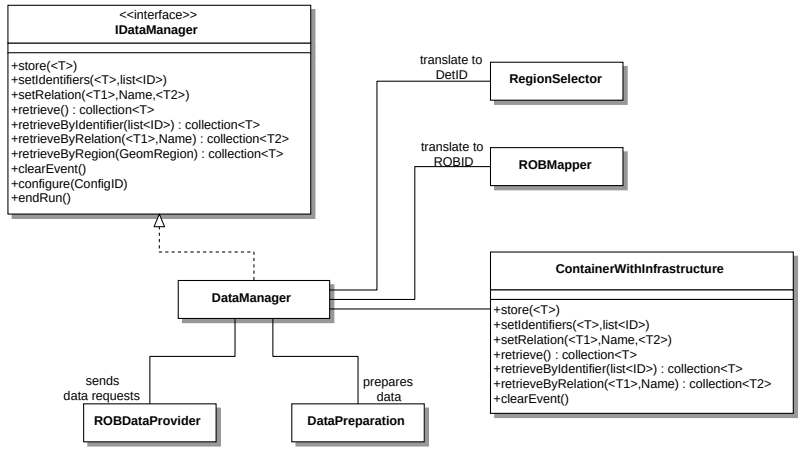


Figure 9-24 The class diagram for the data manager sub-package. See text for details. *THIS NEEDS MAJOR REWORK!!! REGION SELECTION, ROB MAPPER GOES, STOREGATE,...*

The Data Manager provides the means of receiving and storing the event data during the trigger processing. It therefore provides the necessary infrastructure for the EDM and for the navigation used for the seeding mechanism. In case of LVL2, the data manager does the communication with the ROB data provider. It thereby hides the online aspects the ROB data access from the HLT algorithm processing, which is essential for the PESA algorithm development model, which is based on the reuse of offline software.

DISCUSS THE PRINCIPLES OF THE LONDON SCHEME HERE.

- OFFLINE INTERFACES

- OFFLINE TO ROB MAPPING

- ROB DATA PROVIDER

- LAZY DATA PREPARATION

REGION SELECTION NEEDS TO BE INTRODUCED HERE.

The access to (prepared) data by a geometrical region is a clear requirement for most of the HLT algorithms that follows directly from the ROI concept. The **Region Selection** implements this detector geometry dependent data access pattern centrally. It can then be used easily by all HLT algorithms.

THE INTERFACE TO THE DATA IS A BIT MORE COMPLEX AS IT IS RDO/RIO LEVEL DETECTOR CONTAINERS. THE DESCRIPTION OF THE INTERFACE IS STILL WRONG IN THE FOLLOWING.

In figure??? a class diagram for the data manager sub-package is shown. The **Data Manager Interface** is implemented by the data manager. It is used by the HLT algorithms and by the steering. In order to provide the necessary functionality the data manager uses several other components:

- The **Store Gate** is part of the ATHENA framework that provides both, the functionality of a transient store and the infrastructure to implement the raw data access via it persistency mechanism.
- *WRITE SOMETHING ABOUT THE **KEY2KEY** NAVIGATION - ANDREW?*
- The **Region Selector** is used to implement the ‘‘retrieve By Region’’ data access pattern. Their task is to translate the abstract geometry region into a set of (offline) **Identifiers** for **Identifiable Collections**. Those are used to access the data.
- *WRITE SOMETHING ABOUT THE RDO,RIO and **CONTAINERS** THAT HOLD THE DATA - HONG?*
- The **Byte Stream Conversion Service** is...

  EXPLAIN ITS FUNCTION
- The **Data Preparation Tools** are used by the byte stream conversion service to process raw data and to create in the case of the EF the raw data objects or to create in the case of the LVL2 the reconstruction input objects. It hides the detector dependent part of the raw data processing from the HLT algorithm.

The region selector and the byte stream conversion service needs to be configured at the beginning of a ‘‘RUN’’ to access meta data and to set up the data conversion tools used for the data preparation. The same limitations as for the steering and HLT algorithms apply to the meta data access by the data manager, especially for LVL2.

9.4.7.1 Storegate as the Transient Event Store

THIS NEEDS MAJOR REWRITE, IT IS THE OLD TEXT FOR THE CWI BEFORE WE DECIDED TO TAKE STOREGATE.

Storegate as the transient event store of ATHENA is used in the data manager for the function of the container with infrastructure.

STOREGATE:

-DATALINKS

-PERSISTENCY

-ONLINE TO OFFLINE MAPPING

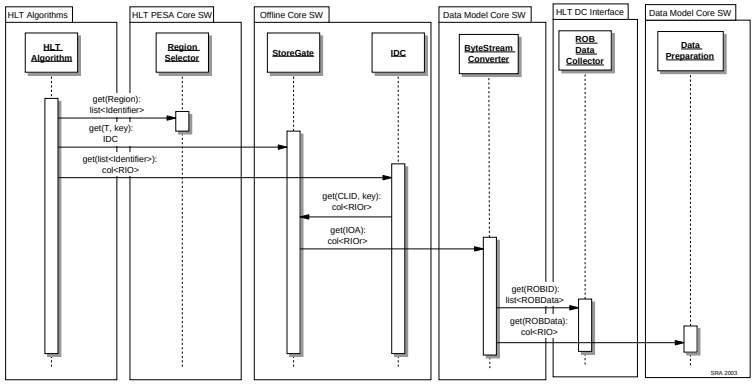
-OWNERSHIP AND CLEANUP

9.4.7.2 The Support for Navigation

ANDREW - ADD SOMETHING ON THE KEY2KEY MECHANISM.

9.4.7.3 The Raw Data Access using the London Scheme

SECTION TO BE WRITTEN UP.



9.4.7.4 Retrieve by Region

STEVE AND ALINE TO REWORK THIS SECTION

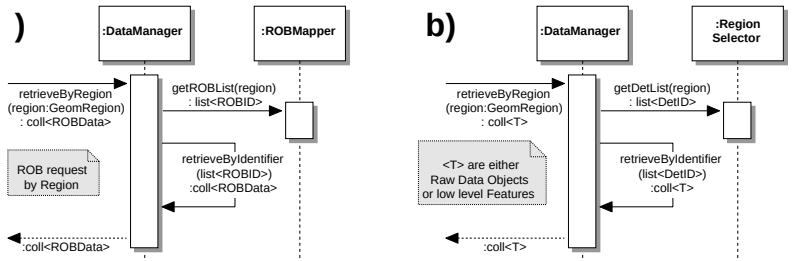


Figure 9-25 Sequence diagrams for the data requests by a geometrical region. The difference between the requests is in the kind of output data. See text for details. *DROP PART A) FROM THE DIAGRAM AND FIX UP PART B.*

The data request by a **Geometrical Region** is an important requirement to implement the ROI concept in the event selection software. A geometrical region does not directly correspond to a ROI, but is a more general concept. A ROI defines a given volume in the detector of interest for the trigger processing, as was shown in figure???. But during the HLT algorithm processing more refined information becomes available that can be used to restrict the original volume and thereby the amount of data to be analysed. On the other hand, a geometrical region could be a full sub-detector or a group of sub-detectors, like for example for the full scan of the inner detector for certain B-physics triggers. Another application of the data request by geometrical region

is for example the “Track Following’’ done by XKALMAN or IPATREC. Here those clusters are to be examined that are on the next detector surface intersected by the track candidate.

In figure??? a sequence diagram is shown...

STEVE - PLEASE ADD TEXT ACCORDING TO THE NEW DIAGRAM.

Note that the “retrieve By Region’’ access pattern effectively hides the details of the detector geometry model and of the Identifiers from the requesting HLT algorithms. This pattern does not apply to higher levels of reconstructed objects, i.e. tracks or TEs. Here the access to these objects is by their relation to other objects (seeding mechanism) in the event.

9.4.7.5 Identifiable Containers and the Reconstruction Input Data

WE NEED TO WRITE HERE SOME DETAILS ABOUT THE CONTAINERS AND THE DATA.

It also supports access to data by **Detector Identifier** for raw data objects or low level features, e.g. calorimeter cells or clusters in the SCT. A Detector Identifier corresponds, for example, to a wafer for the Pixels \cite{IDEDMReq}.

9.4.7.6 Data Request by Detector Identifiers

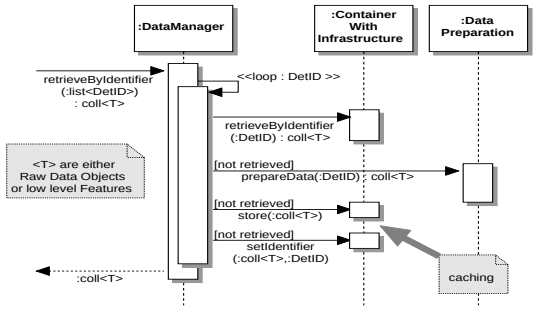


Figure 9-26 A sequence diagram for the data request by detector identifier. See figure??? for details of the data preparation. *UPDATE FIGURE FOR THE LONDON SCHEME IMPLEMENTATION. HENCE, IDC, BYTESTREAM-ADDRESS/PROXY, PERSISTENCY.*

THIS TEXT NEEDS REWORKING FOR THE LONDON SCHEME.

In figure??? the corresponding sequence diagram is shown for the data request by detector identifier. Again, the data manager checks, if the data for a given detector identifier is available in the Storegate. If not, the data preparation is used to prepare the requested output, because either raw data objects or low level features are requested. Again, the data is then stored in Storegate to allow for caching before returning it.

9.4.7.7 ROB Data Request and Lazy Data Preparation

UPDATE THIS TEXT FOR THE LONDON SCHEME.

In figure??? a sequence diagram is shown for the data preparation. It prepares raw data objects or low level Features if they are not already stored in Storegate, as was shown in figure???. This way the important performance requirement of so called ‘‘lazy data preparation’’ is implemented, because only the requested part of the data is actually processed, when it is needed by the HLT algorithm.

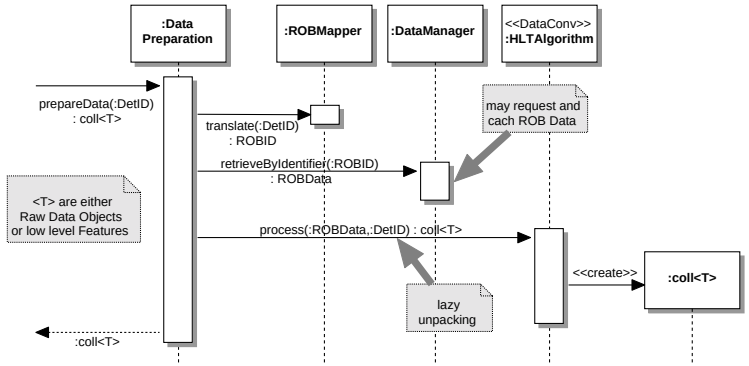


Figure 9-27 A sequence diagram for the data preparation that executes the data conversion algorithms inside the data manager. *REWORK - ROB DATA ACCESS AND BYTE STREAM CONVERTER.*

THIS TEXT BELOW IS WRONG, NEEDS UPDATE.

As shown in the figure, the data preparation uses the ROB mapper to translate the detector identifier into a ROB Identifier. Then the ROB Data is requested from the Data Manager itself. This request was discussed in figure???. The Data Preparation uses a Data Conversion Algorithm to create the requested output data from the part of the ROB Data that corresponds to a given Detector Identifier. It is an option, that only a part of the full ROB Data fragment gets transmitted from the ROS.

9.4.8 Further Issues

THIS IS A PLACEHOLDER FOR MORE STUFF TO COME. THE FOLLOWING STUFF NEEDS PROPER WRITE-UP FOR THE NEXT DRAFT:

- WRITE SOMETHING ON THE ISSUES OF BUNDLED ROB REQUESTS.
- An issue that needs to be addressed by the detailed design and implementation are HLT algorithm error conditions. They are to be handled by the sequencer and by the data preparation.
- The LVL2 supervisor or the PESA steering controller may timeout events that are being processed in the LVL2. In such a case the LVL2 result might not be send to the ROS and might therefore not be available at input to the EF. Another possibility is that the event selection software sends only a partial result because of an error condition. In both situations the EF might be forced to revert back to the LVL1 result to seed the selection, which needs to be taken into account in the detailed design of the EF selection strategy.

- The meta data services are needed to provided the necessary information for the event selection software. An important issue here is the lifetime of the information. In the LVL2 all meta data accesses are to be done before the start of the ‘‘RUN’’, while for the EF updates per time period may be possible.
- The monitoring has several aspects, the debug output of the HLT algorithms, the timing, the rate monitoring, etc.. Monitoring services are used by the event selection software to publish the information.
- It is desirable to allow for a graphical event display in order to visualise the trigger reconstruction and decision making process. This could be done on the basis of the event information transmitted via the LVL2 and EF results that may include all reconstructed Features for debugging purposes.
- For the online system a graphical user interface will be useful for both the control and the monitoring of the event selection software.

9.5 References

9-1 LVL2 URD

[1] F. Touchard et al., HLT operational analysis and requirements to other sub-systems, http://edms.cern.ch/...
[2] C. Bee et al., Event Handler Requirements, http://edms.cern.ch/...
[3] C. Meessen et al., Event Handler Functional Design Analysis, http://atlasinfo.cern.ch/Atlas/GROUPS/DAQTRIG/EF/EFD-FunctionalAnalysis.pdf
[4] C. Bee et al., Event Handler Design, http://edms.cern.ch/...
[5] C. Bee et al., Supervision requirements, http://edms.cern.ch/...
[6] C. Bee et al., ATLAS Event Filter: Test Results for the Supervision Scalability at ETH Zurich, November 2001, http://documents.cern.ch/cgi-bin/setlink?base=atlnot&categ=Note&id=daq-2002-006
[7] S. Wheeler et al., Test results for the EF Supervision, https://edms.cern.ch/document/374118/1

9-2

9-3 EF DataFlow URD

9-4 EF Supervision URD

9-5 ESS Requirements Doc

9-6 ESS Design Doc

9-7 Guidelines for writing thread-safe LVL2 algorithms

9-8 Athena

9-9 Gaudi

9-10 PSC

9-11 April prototype

10 Online Software

10.1 Introduction

The Online Software encompasses the software to configure, control and monitor the TDAQ system but excludes the management, processing and transportation of physics data. It is a customizable framework which provides essentially the "glue" that holds the various sub-systems together. It does not contain any elements that are detector specific as it is used by all the various configurations of the TDAQ and detector instrumentation. It co-operates with the other sub-systems and interfaces to the Detector Readout Crates, the Detector Control System (DCS), the Level 1 Trigger, the Data-flow, the High Level Trigger processor farms, the Offline Software and to the human user as illustrated in Figure 10-1.

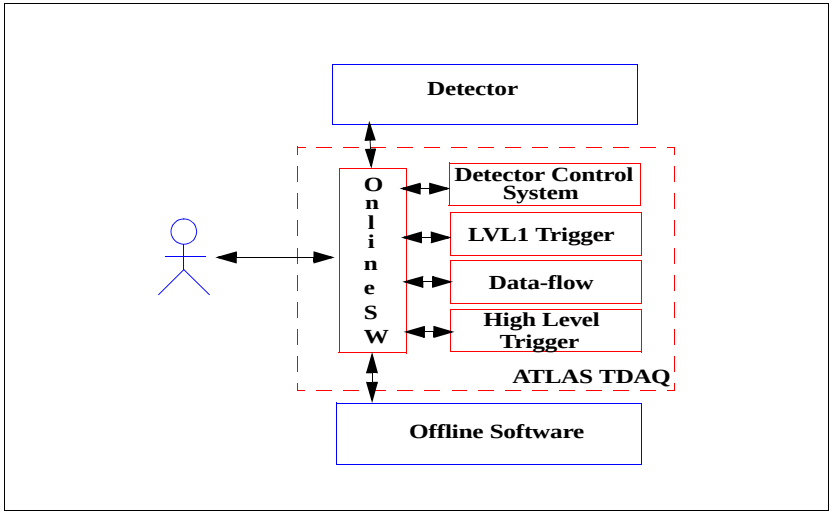


Figure 10-1 The Online Software in relation to the TDAQ system

An important task of the Online Software is to provide services to marshal the TDAQ through its start-up and shutdown procedures so that they are performed in an orderly manner. It is responsible for the synchronisation of the states of a run in the entire TDAQ system and for process supervision. These procedures are aimed to take a minimum amount of time to execute to reduce the overhead since this affects the total amount of data that can be taken during a data-taking period. Verification and diagnostic facilities help for early and efficient problem finding. Configuration database services are provided for holding the large number of parameters which describe the system topology, hardware and software components and running modes, based on the partitioning model. During data taking, access to monitoring information like statistics data, sampled data fragments to be analysed by monitoring tasks, histograms produced in the TDAQ system, and also to the errors and diagnostic messages sent by different applications is provided. User interfaces display the status and performance of the TDAQ system and

allow the user to configure and control his operation. These interfaces provide comprehensive views of the various sub-systems at different levels of abstraction.

The Online software has various types of users. The *TDAQ Operator* runs the *TDAQ system* in the control room during a data-taking period, the *TDAQ Expert* has system-internal knowledge and can perform major changes to the configuration, the *Sub-system or Detector Expert* is responsible for the operation of a particular sub-system or detector and *TDAQ and detector software applications* use services provided by the Online software via application interfaces.

remark: the following definitions will be available from the TDAQ glossary to which only a reference will be made.

The following types of Online Software users have been identified:

1. *TDAQ Operator - this user is responsible for using the TDAQ system to take data during a particular data taking session, for example during a shift. He has to be able to start, monitor and stop data taking. He is not expected to perform any programming tasks related to the Online Software.*
2. *TDAQ Expert - this user is responsible for running and maintaining the TDAQ system itself. He is responsible for the initialisation and shutdown of the TDAQ system or parts of it. He shall have a knowledge of the TDAQ structure and its components.*
3. *TDAQ Sub-System or Detector Expert - this user is responsible for the operation of a particular sub-system of the TDAQ or particular sub-detector of the ATLAS detector. He should be capable of describing the specific TDAQ sub-system or detector configuration and diagnosing the specific sub-system or detector problems which may appear during operation.*
4. *TDAQ Sub-System or Detector - this user represents a software produced by the TDAQ Sub-System or Detector developers. This software will use the services provided by the Online Software for the sub-systems and detectors configuration and control.*

The user requirements to the Online Software are collected and described in the corresponding document [10-1].

10.2 The Architectural Model

The Online Software architecture is based on a component model and consists of three high-level components, called packages. Details on the architecture and a comprehensive set of use cases are described in [10-2]. Each of the packages is associated with a functionality group of the Online software. For each package a set of services which it has to provide is defined. The services are clearly separated one from another and have well defined boundaries. For each service a low-level component, called sub-package, is identified.

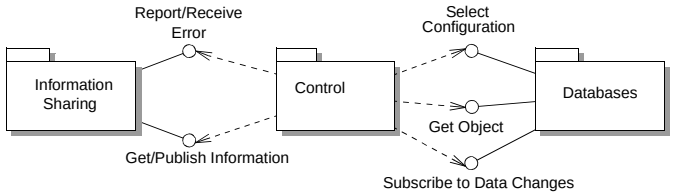
Each package is responsible for a clearly defined functional aspect of the whole system.

1. Control - contains sub-packages for the control of the TDAQ system and detectors. Control sub-packages exist to support TDAQ system initialisation and shutdown, to provide control command distribution, synchronisation, error handling and system verification.
2. Databases - contains sub-packages for configuration of the TDAQ system and detectors. Configuration sub-packages exist to support system configuration description and access to it, record operational information during a run and access to this information. There are also boundary classes to provide read / write access to the conditions storage.

3. Information Sharing - contains classes to support information sharing of the TDAQ system and detectors. Information Sharing classes exist to report error messages, to publish states and statistics, to distribute histograms built by the sub-systems of the TDAQ system and detectors and to distribute events sampled from different parts of the experiment's data flow chain.

The Internal Interaction between the Online Software Packages is shown in Figure 10-2. The Control makes use of the Information Sharing and of the Databases packages. The Databases package is used to describe the system to be controlled. This includes in particular the configuration of TDAQ Partitions, TDAQ Segments and TDAQ Resources. The Information Sharing package provides the infrastructure to obtain and publish information on the status of the controlled system, to report and receive error messages and to publish results for interaction with the operator.

Figure 10-2 Internal Interaction between the Online Software Packages



The following sections describe these packages in more details.

10.3 Control

The main task of the control package is to provide the necessary tools to perform the TDAQ system operation as they are described in Chapter 3. It provides the functionality of the TDAQ Control as shown in the controls view of the Chapter 5.

In addition the package has the responsibility for functionalities necessary in the computing environment for user interaction, process management and access control.

10.3.1 Control Functionality

Control encompasses software components responsible for the control and supervision of other TDAQ systems and the detectors. The functionalities have been derived from the user requirements and are described in turn.

- **User Interaction:** Interaction with the human user like the operator or system expert of the TDAQ system
- **Initialization and shutdown:** Initialisation of TDAQ hardware and software components is foreseen. The operational status of system components must be verified and the initialisation of these components in the required sequence is ensured. Similar considerations are required for the shutdown of the system.

- **Run control:** System commands have to be distributed to a range of several hundred to thousand of clients programs. The control sub-package is responsible for the command distribution and the required synchronisation between the TDAQ sub-systems and detectors.
- **Error handling:** Malfunctions can interrupt system operations or deteriorate the quality of physics data. It is the task of the control sub-package to identify such malfunctions. If required the system will then autonomously perform recover operations and assist the operator with diagnostic information.
- **Verification of System status:** The control package is responsible to verify the functionality of TDAQ configuration or any subset of it.
- **Process Management:** Process management functionality in a distributed environment is provided.
- **Resource Management:** Management of shared hardware and software resources in the system is provided.
- **Access Management:** The control package provides a general Online software safety service, responsible for TDAQ user authentication and the implementation of an access policy for preventing non-authorised users to corrupt TDAQ functionality.

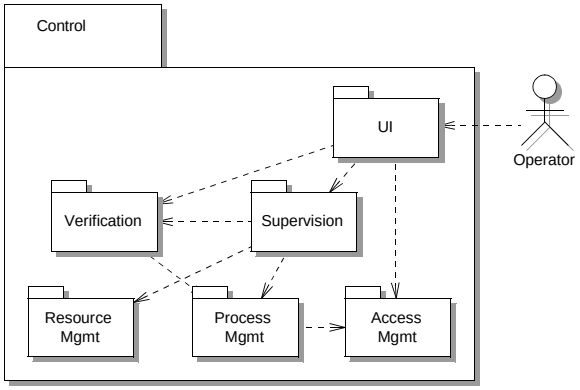
10.3.2 Performance and Scalability Requirements on Control

The TDAQ system is a large and heterogeneous system composed out of a large number of items to be controlled. These items range from read-out modules in VME crates to workstations within HLT computer farms. Typically these items are clustered such that modules are contained within crates or workstations are part of sub-farms. Such units are the preferred places to interface with the Online Software Control system. The number of these units is estimated to be in the range of 500 to 2000. To control these units the TDAQ control system is build in a hierarchical and distributed manner. More detailed explanation can be found in the chapter on Experiment Control, Chapter 12.3.1, "Online Software Control Concepts".

10.3.3 Control Architecture

The Control package is divided into a number of sub-packages as shown in Figure 10-3.

Figure 10-3 The Organization of the control package



The functionality described in Section 10.3.1 has been distributed to several distinct sub-packages:

- The User Interface (UI) 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 authorisation when necessary

10.3.3.1 User Interface

The **User Interface** (UI) provides an integrated view of the TDAQ system to the operator and should be the main interaction point. It is foreseen to provide a flexible and extensible UI that can accommodate panels implemented by the detectors or TDAQ systems. Web based technologies will be used to give access to the system for off site users.

10.3.3.2 Supervision

The **Supervision** sub-package realizes the essential functionality of the Control package. The generic element is the so-called controller. 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 via its higher level controller. One top level controller

called root controller will take the function of the overall control and coordination of the system. by interfacing to other TDAQ system or detector specific controllers. Via the User Interface sub-package it provides all TDAQ control facilities to the Operator. These are expressed as interfaces in Figure 10-4 and discussed below in more details.

The *Initialisation and Shutdown* is responsible for

- initialization of TDAQ hardware and software components, bringing TDAQ partition to the state when it can accept Run commands.
- re initialization of a part of the TDAQ partition when necessary
- 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
- analysing the status of controlled sub-systems and presenting the status of the whole TDAQ to the Operator

The *Error Handling* is concerned with

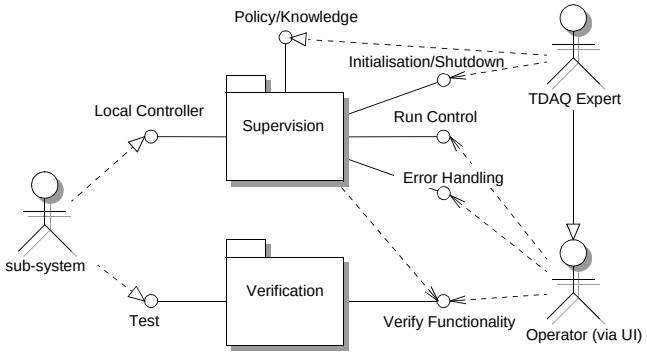
- analysing run-time error messages coming from TDAQ sub-systems
- diagnosing problems, proposing recovery actions to the operator or performing automatic recovery if requested

Most of the above defined functionality can reside on the local controller and are extended by specific policies which the TDAQ sub-systems and detector expert developers implement.

10.3.3.3 Verification

The **Verification** sub-package is responsible for the verification of the functionality of the TDAQ system or any subset of it. It uses developer's knowledge to organize tests in sequences, analyse test results, diagnose problems and provide a conclusion about the functional state of TDAQ components

Figure 10-4 Interfaces of the Supervision and Verification sub-packages



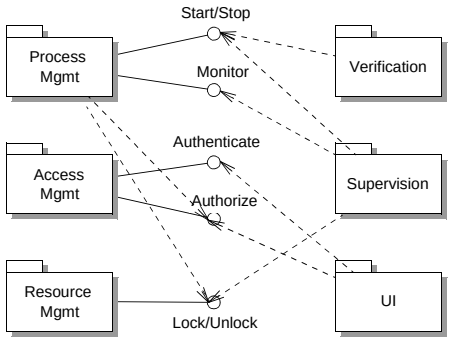
A TDAQ sub-system developer implements and describes tests which are used to verify any SW or HW component in a configuration. This includes also complex test scenario, where the component functionality is verified by the simultaneous execution of processes on several hosts. The sub-system uses the Process Management sub-package for the execution of tests.

The Verification sub-package is used by the Supervision to verify the state of the TDAQ components during initialization or recovery operations. It can also be used directly by the Operator via UI, as it is shown on Figure 10-4.

10.3.3.4 Process, Access and Resource Management systems

The Verification and Supervision sub-packages connect via interfaces to other Control sub-packages, as shown in Figure 10-5.

Figure 10-5 Interfaces of the Process management, resource Management and the Access Management



The **Process Management** provides basic process management functionality in a distributed environment. This functionality includes starting, stopping and monitoring processes on different TDAQ hosts.

The **Resource Management** is concerned with the allocation of software and hardware resources between running partitions. It prevents the operator from performing operations on resources which are allocated to other users.

The **Access Management** is a general Online software safety service, responsible for TDAQ user authentication and implementation of an access policy, in order to disable non-authorised persons to corrupt eventually TDAQ functionality.

10.3.4 Prototype Evaluation

Prototype evaluations have been performed for a number of technologies. The initial choice was based on experiences in previous experiments. Products were chosen, that fit well in the envisaged object oriented software environment.

- A Run Control implementation is based on a State Machine model and uses the State machine compiler CHSM as underlying technology.
- A Supervisor is mainly concerned with process management. It has been build using the Open Source expert system CLIPS.
- A verification system, DVS, performs tests and provides diagnosis. It is equally based on CLIPS.
- A Java based graphical User Interface, IGUI is in use.
- Process Management and Resource Management are implemented based on components which are part of the current implementation of the Online Software packages.

10.3.4.1 Scalability and Performance Tests

A series of large scale performance tests has been performed to investigate the behaviour of the prototype on systems of the size of the final TDAQ system. In several iterations the behaviour of the system was studied and limits were identified, the prototype was refined and tested until the successful operations of a system with a size in required range was reached. Up to 1000 controllers and their applications could be controlled reliably while meeting the performance requirements.

One item of interest is the synchronized distribution of commands within the system. A command is send by the root controllers and propagates through the controller tree to the leaf controllers. These leaf controllers interface from the overall Online control system to the specific tasks to be performed in the TDAQ system. The time was measured to distribute a command from the top level and to obtain the confirmation from all controllers. Figure 10-6 presents the time taken to perform three state transition in turn for a three level control hierarchy for configurations containing 10 to 1000 leaf controllers. A single transition takes about 2 seconds for 1000 controllers, a time well within expectations. In a real world system each controller would perform its specific actions during such a transition. This would need a time between a few seconds up to tens of seconds. In relation to these values the overhead of the overall control system will be small.

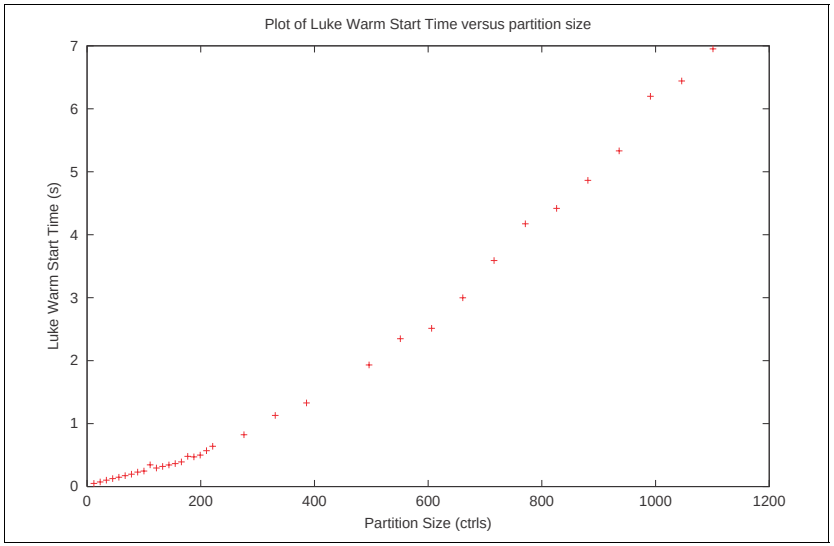


Figure 10-6 Time taken to perform three state transitions for configurations containing 10 to 1000 leaf controllers.

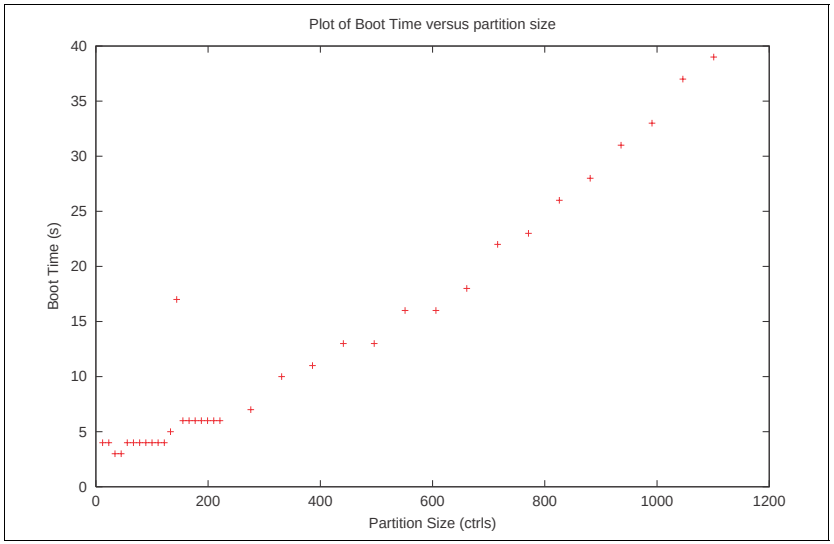


Figure 10-7 Boot transition in a three level control hierarchy for configurations containing 10 to 1000 leaf controllers. To initialize a system with thousand controllers currently 30 seconds are needed.

A second item of interest is the process control in the distributed system. To switch between various configurations, it will be necessary to start and stop many processes on many nodes. While such an operation is not performed during physics data-taking, it will be of importance during development and calibration. Figure 10-7 shows the time needed by the prototype implementation to initialize a system with thousand controllers. Detailed descriptions and further test results can be found in [10-3].

10.3.4.2 Technology Considerations

During the evaluation of the prototype several shortcomings of the current system have been identified. An important aspect is the lack of flexibility in the state machine implementation CHSM. Another aspect is the multitude of applied technologies. It is envisaged that the final supervision system will be based on a single technology. In this context the expert system CLIPS and related products have been studied. Due to the general purpose nature of this product various aspects of the supervision like initialization, control and error handling can well be implemented. The knowledge base provides the basis for customizable solutions, that can be specialized for different parts of the system. Another advantage is the extensibility of CLIPS. It can easily be interfaced with other components of the Online Software system. Alternative products in this category are Jess, a similar expert system implementation written in Java and a commercial alternative, Eclise by Haley. Inc.

Alternatives have been investigated, for example solutions based on scripting languages. The use of SMI++, a scripting language for the description of interaction state machines, which is used by several HEP experiments, has been studied, as well as a possible implementation based on the scripting language Python. While each of these approaches has its particular merits, the evaluation showed that the CLIPS based solution is better suited for our environment and is the favoured implementation choice.

10.4 Databases

The TDAQ systems and detectors require several persistent services to access description of the configuration used for the data taking as well as to store the conditions under which the data were taken. The online software provides common solutions for such services taking into account requirements coming from the TDAQ systems and detectors.

10.4.1 Functionality of the Databases

There are three main persistent services proposed by the online software:

- the **configuration databases** to provide the description of the system configurations,
- the **online bookkeeper** to log operational information and annotation,
- the **conditions databases interface** to store and read conditions under which data were taken.

10.4.1.1 Configuration Databases

The configuration databases are used to provide overall description of the TDAQ systems and detectors hardware and software. It includes description of partitions defined for the system and parameterized for different types of runs describing the hardware and software used by a given partition and their parameters.

The configuration databases are organized in accordance with the actual hierarchy of the TDAQ system and detectors. The configuration databases give a possibility for each TDAQ system and detector to define their own format of the data (i.e. the database schema), to define the data themselves and to share the schema and the data with others. The configuration databases provide graphical user interfaces to browse the data by any human user and to modify the data by authorized human experts. The configuration databases give possibility to generate data access libraries which hide the technology used for the databases implementation and can be used by any TDAQ or detector application to read the configuration description or to be notified in case of it's changes. An application started by the authorized expert can use the data access libraries to generate or to modify the data.

The configuration databases provide efficient mechanism for fast access to the data for huge number of clients during data taking. They do not store the history of the data changes but provide archiving options. Configuration data which are important for offline analysis must be stored in the conditions databases.

10.4.1.2 Online Bookkeeper

The online bookkeeper is the system responsible for the online storage of relevant operational information and configuration description provided by the TDAQ systems and detectors. The OBK organizes the stored data on a per-run basis and provides querying facilities.

The online bookkeeper provides graphical user interfaces to allow human users to browse contents of the recorded information or append such information. The append access is limited for authorized users only. Similarly, the online bookkeeper provides application programming interfaces for user applications to browse the information or to append annotations.

10.4.1.3 Conditions Databases Interfaces

The TDAQ systems and detectors use the conditions databases to store conditions which are important for the offline reconstruction and analysis. The conditions data are varying with time and physical parameters such as temperature, pressure and voltage. These conditions are stored with a validity range (typically time or run number) and retrieved using time or run number as a key.

The conditions databases are implemented and supported by the offline software group. The online software group provides application programming interfaces for all TDAQ systems and detector applications and mechanisms to insure required performance during data taking runs.

10.4.2 Performance and Scalability Requirements on the Databases

The performance and scalability requirements to the database services are strongly depend on the strategies chosen by the TDAQ systems and detectors to get the data to their applications, to

store the conditions and to keep the operational data. Roughly, for most of TDAQ systems and detectors, the number of clients of configuration and conditions database can be equal to the number of local controllers, which varies by different estimations from 500 to 2000. For EF the situation is not defined yet and the number of database clients in the worse scenario can be $O(10^3)$, if each processing task will get configuration description and conditions itself.

The databases information can be split on several not crossed domains specific for the TDAQ systems and detectors. The complete description is required only to few applications, while the most others require to access only a small fraction of it. The typical database size which completely describe all necessary parameters of TDAQ system or detector for a physics run is about $O(10^2)$ MBytes. The DCS may produce up to 1 GBytes of measured conditions per day.

The databases data are read once during starting of the data taking and an acceptable time to get description for whole system is an order of few minutes. During the data taking of physics data the databases may slightly be changed and an acceptable time to get the changes by registered applications is several seconds. The data can be considerably changed during special calibration runs and the maximum rate requested in this case is 10 Mbytes produced in one hour.

10.4.3 Architecture of Databases

10.4.3.1 Configuration databases

The configuration databases (ConfDB) provide user and application programming interfaces.

Via user interface the software developer defines the data object model (i.e. the database schema) describing required configurations. The expert uses the interface to manage databases, to put the system description and to define configurations, which can be browsed by a user.

A TDAQ or detector application access databases via data access libraries (DAL). A DAL is generated by the software developer for a part of the defined object model relevant to his sub-system. The DAL is used by any process required to get the configuration description or to receive a notification in case of it's changes. Also, the DAL is used by an expert process needed to produce the configuration data.

The ConfDB system contains the following classes:

- **ConfDB Repository** - defines low-level interfaces to manipulate the configuration databases including databases management by users which are granted enough privileges, schema and data definitions and notification subscription on data changes; it defines interfaces above a DBMS used for implementation and hides any specific details, so any other ConfDB classes shall never use DBMS-specific methods directly;
- **ConfDB UI** (User Interface) - defines user interface for object model definition, configurations definition, database browsing and population by human actors;
- **ConfDB DAL Generator** - defines interface to produce DAL;
- **ConfDB DAL** - defines interfaces for configurations selection, reading/writing configuration objects and subscription for notifications on data changes by the user and expert processes (to receive notification a process shall realize **ConfDB Data Monitor** interface).

The Figure 10-1 shows interfaces provided by the configuration databases and their users.

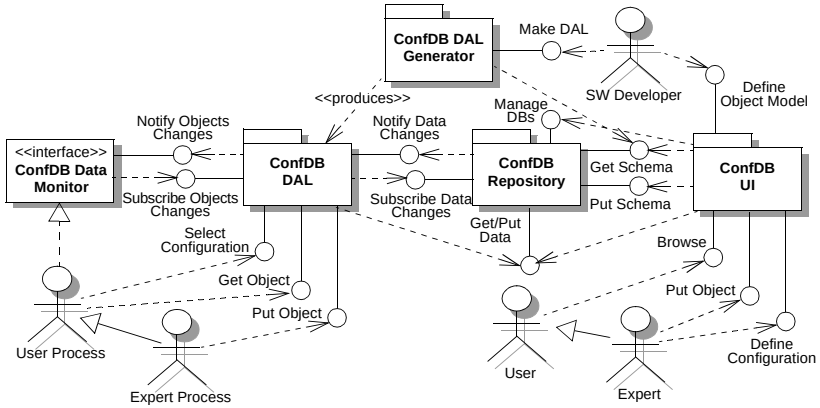


Figure 10-8 Configuration databases users and interfaces

10.4.3.2 Online bookkeeper

The OBK provides several interfaces to its services, being that some of them are human oriented, while others are APIs to allow interfacing with client applications. The access to these interfaces depends on the user's (human or system actor) privileges.

The OBK uses as persistency backbone the Conditions and Time-Varying offline databases services. In this sense, it counts on those services to provide the necessary means to store and retrieve coherently data that changes in time and of which there may exist several versions (e.g. configurations). In Figure 10-2 it is possible to observe the logical subdivision of the OBK system into abstract classes. Of these, the main ones are:

- **OBK Repository** - defines the basic methods to allow the storing, modifying and reading of online log data, as well as the methods to set the OBK acquisition mode and also to request log data from the several TDAQ processes providing them. It allows a human or system actor to start or stop the acquisition of log data. In order to become a log data provider a TDAQ application will have to realize the **OBK Log Information Provider** interface. This interface allows a TDAQ application to accept subscriptions for log information from the OBK, as well as for the OBK to access log information in a TDAQ application;
- **OBK Browser** - this is the class responsible for providing the necessary querying functionality for the OBK database. Since the data browsing and data annotation functions are tightly coupled, the class also includes functionality to add annotations to the database;
- **OBK Administrator** - the OBK Administrator class provides to the users which are granted enough privileges the functionality to alter (delete, move, rename) parts or all of the OBK database. These users also given the possibility of changing the OBK acquisition mode (e.g. data sources, filters for the data sources).

Apart from the main classes depicted in Figure 10-2, OBK's architecture also includes four other classes (not shown in the diagram for reasons of clarity): **OBK UI Browser** and **OBK Browser API** both inherit from the OBK Browser class and define the human client oriented and the application client oriented versions of that class; the same thing happens for the **OBK UI Administrator** and the **OBK Administrator API** classes which defines the human client and application client oriented versions on the OBK Administrator class.

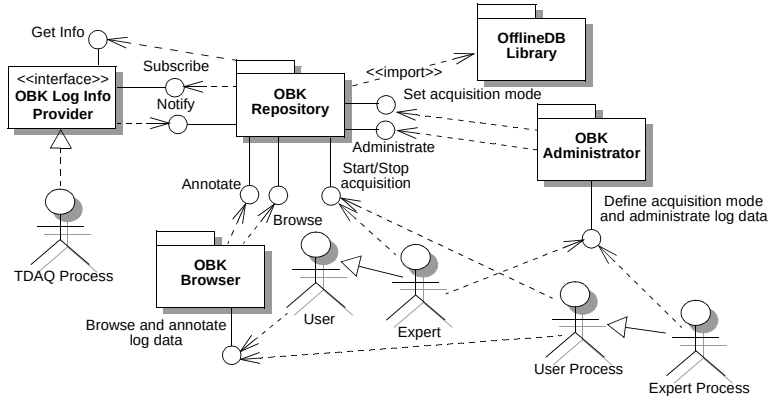


Figure 10-9 OBK users and interfaces

10.4.3.3 Conditions database interface

The user requirements to the ATLAS offline conditions and time-varying databases and their architecture are not specified at the moment of the document writing. The conditions database interface will add functionality required by TDAQ and detectors users, if it will not be provided by the offline.

10.4.4 Application of databases to the TDAQ sub-systems

Usage of the databases by the other TDAQ systems, concentrating on differences with general use.

Should be provided by TDAQ systems

10.4.5 Prototype evaluation

The main required functionalities of the ConfDB and the OBK have been implemented and evaluated in the prototype of the Online System. The main goals were to have a prototype to be used during test beams and integration with other TDAQ systems and detectors, and to proof the possibility to use chosen technologies for the final system.

10.4.5.1 Configuration Databases

The prototype of the configuration databases is implemented on top of the OKS persistent in-memory object manager. It allows to store the database schema and data in multiple XML files

which subset can be combined to get a complete description. The access to the configuration description can be made via file system (C++ interface) and via dedicated remote database server built on top of ILU (freeware CORBA implementation) and tested for C++ and Java interfaces.

The Figure 10-10 presents results obtained during online software performance and scalability tests [REF]. The possibilities to read a realistic configuration via AFS file system and from remote database server were tested for maximum available amount of nodes.:

Figure 10-10 The results of the configuration databases performance and scalability tests

The tests with direct access of the configuration via common AFS file system had shown good scalability and performance and such approach can be used if a common file system will be available. The results of access via remote database server can indicate the number of the servers which need to be started depending on number of clients, requests synchronisation, amount of data they read and time requirements.

The proposed architecture of the configuration databases allows to switch between implementation technologies without affecting user code. Some other database technologies are studied for possible replacement of the one used in the prototype including relational databases (MySQL and ORACLE) with possible object extensions and POOL LCG project.

10.4.5.2 Online Bookkeeper

The prototype of the online bookkeeper was implemented on OKS persistent in-memory object manager and MySQL freeware implementation of relational database management system. The results obtained during recent performance and scalability tests [REF] have shown, that the current MySQL implementation allows to reach 20 KBytes per second rate when storing monitoring data (100 bytes per data item) produced by up to 100 providers.

10.5 Information Sharing

The choice of name for this section is not final. A possible alternative could be "Monitoring services". This would then be applied to the whole of the document where one talks about these services.

There are several areas where information sharing is necessary in the TDAQ system: synchronisation between processes, error reporting, operational monitoring, physics event monitoring, etc. The Online Software provides a number of services which can be used to share information between TDAQ software applications. This chapter will describe the architecture and prototype implementation of these services.

10.5.1 Functionality of the Information Sharing Services

Any TDAQ software application may produce information which is of interest for the other TDAQ applications. The first application will be called in this chapter Information Provider, the later ones will be called Information Consumers, which indicates that they are users of the information. Any TDAQ software application may be Information Provider and Information Consumer at the same time. The main responsibility of the Information Sharing services is:

- transportation of the information from the Information Providers to the Information Consumers
- delivery of information requests from the Information Consumers to the Information Providers.

Figure 10-11 shows main interactions which providers and consumers may have with the Information Sharing services.

Figure 10-11 Information Sharing in the TDAQ system

10.5.2 Performance and scalability requirements on Information Sharing

It is expected that the TDAQ system will contain about $O(10^3)$ processes. Each of those processes can produce information of different types. Therefore each Information Sharing service shall be able to serve $O(10^3)$ Information Producers simultaneously.

The number of Information Consumers for any single information item is expected to be about O(1) processes. Therefore each Information Sharing service shall be able to serve O(1) Information Consumers of each information item simultaneously.

The time to transport a single information object from the Information Provider to all the interested Information Receivers shall be about O(1) milliseconds.

10.5.3 Architecture of Information Sharing Services

The Online Software provides four services to handle different types of shared information. Each service offers the most appropriate and efficient functionality for a given information type and provides specific interfaces for both Information Providers and Consumers. Figure 10-12 shows the existing services.

The Inter Process Communication (IPC) is a basic communication service which is common for all the other Online Software services. It defines high-level API for the distributed object implementation and for remote object location. Any distributed object in the Online Software services

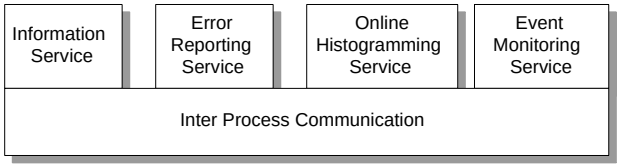


Figure 10-12 Information Sharing services

has common basic methods which are implemented in the IPC. In addition the IPC implements partitioning, allowing to run several instances of the Online Software services in different TDAQ Partitions concurrently and fully independently.

10.5.3.1 Information Service

The Information Service (IS) allows software applications to exchange user-defined information. Figure 10-13 shows interfaces provided by the IS.

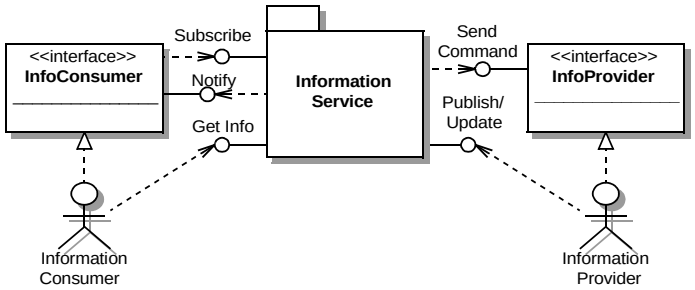


Figure 10-13 Information Service interfaces

Any Information Provider can make his own information publicly available via the Publish interface. Then there are two possibilities. The Information Provider, which does not implement the InfoProvider interface, has to inform the IS about all the changes of the information via the Update interface. The Information Provider, which implements the InfoProvider interface, updates the information only when it is explicitly requested by the IS via the Send Command interface.

There are also two types of Information Consumers. One can access the information by request via the Get Info interface. This one does not need to implement the InfoConsumer interface. The Information Consumer, which implements the InfoConsumer interface, is informed about changes of the information, for which it subscribed via the Subscribe interface.

10.5.3.2 Error Reporting Service

The Error Reporting Service (ERS) provides transportation of the error messages from the software applications which detect these errors to the applications which are responsible for their handling. Figure 10-14 shows interfaces provided by the Error Reporting Service.

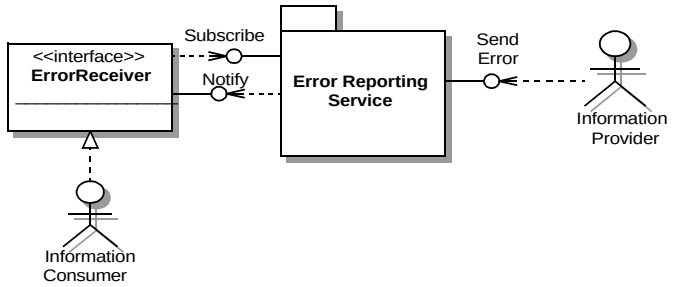


Figure 10-14 Error Reporting Service interfaces

An Information Provider can send the error message to the ERS via the Send Error interface. This interface can be used by any application which wants to report an error. In order to receive the error messages an Information Consumer has to implement the ErrorReceiver interface and construct the criteria which defines what kind of messages it wants to receive. This criteria has to passed to the ERS via the Subscribe interface.

10.5.3.3 Online Histogramming Service

The Online Histogramming Service (OHS) allows applications to exchange histograms. The OHS is very similar to the Information Service. The difference is that the information which is transported from the Information Providers to the Information Receivers has pre-defined format. Figure 10-15 shows interfaces provided by the Online Histogramming Service.

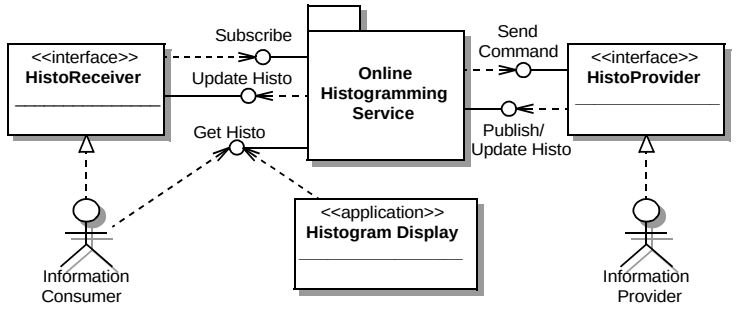


Figure 10-15 Online Histogramming Service interfaces

The OHS sub-package will provide also a human user interface in a form of an application. This application is called Histogram Display and can be used by the TDAQ operator to display available histograms.

10.5.3.4 Event Monitoring Service

The Event Monitoring Service (EMS) is responsible for transportation of physical events or event fragments sampled from well-defined points in the data flow chain to the software applications which can analyse them in order to monitor the state of the data acquisition and the quality of physics data of the experiment. Figure 10-16 shows main interfaces provided by the Event Monitoring Service.

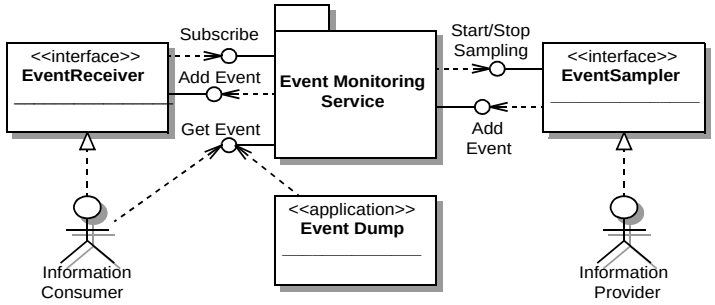


Figure 10-16 Event Monitoring Service interfaces

The application which is able to sample events from a certain point of the data flow has to implement the Event Sampler interface. When the Information Consumer requests the samples of events from that point, the EMS system ask the Information Provider via the Start Sampling interface to start sampling process. The Information Provider samples events and provides them to the EMS via the Add Event interface. When there are no more Information Consumers interesting in event samples from that point of the data flow chain, the EMS system ask the Information Provider via the Stop Sampling interface to stop sampling process.

There are also two types of interfaces for the Information Consumer. One is a simple Get interface which allows consumer to ask event samples one by one when they become necessary. This interface will be used for example by the Event Dump application that implements a human user interface to the EMS system. A second interface is based on the subscription model. Using it the Information Consumer can ask the EMS system to supply the samples of event as soon as they are sampled by the Information Provider. This interface is more suitable for the monitoring tasks which need to monitor events for a long time in order to accumulate the necessary statistics.

10.5.4 Application of Information Sharing Services to the TDAQ sub-systems

Usage of the information services by the other TDAQ systems, concentrating on differences with general use.

Should be provided by TDAQ systems

This sub-section should only exist if the information is not already covered in Chapter 7, "Monitoring".

10.5.5 Prototype evaluation

The prototype implementations have been done for all the Information Sharing services. These prototypes are aiming to proof the feasibility of the chosen design and implementation technology for the final TDAQ system, and to be used for the ATLAS test beams. This chapter contains description of the services implementation along with their performance and scalability test results.

10.5.5.1 Description of the Current Implementation

The Online Software provides prototype [10-4] implementations for all the Information Sharing services. Each service is implemented as a separate software package with both C++ and Java interfaces. All the services are partitionable in a sense that it is possible to have several instances of each service running concurrently and fully independently in different TDAQ partitions.

The Information Sharing services implementation is based on the Common Object Request Broker Architecture (CORBA) defined by the Object Management Group (OMG). CORBA is a vendor-independent specification for an architecture and infrastructure that computer applications use to work together over networks. The most important features of the CORBA are: object oriented communication, inter-operability between different programming languages and different operating systems, object location transparency.

10.5.5.2 Performance and scalability of current implementation

The most exhaustive tests have been done for the Information Service which provides the most general facility for the information sharing. The other services are implemented on the same technology and will offer the same level of performance and scalability as the IS.

The test bed for the IS test consists from 216 dual-pentium PCs with processor frequency from 600 to 1000 MHz. [10-3] The IS has been set up on one dedicated machine. Another 200 machines have been used to run from 1 to 5 Information Providers on them. Each Information Provider publishes one information object at the start and then updating it once per second. The last 15 machines were used to run 1, 5, 10 or 15 Information Consumers which subscribe for all the information in the IS. Whenever an Information Provider changes the information, this new information was transported to all the Information Consumers.

The time for transporting information from one Information Provider to all the subscribed Information Consumers have been measured. Figure 10-17 shows the average of the measured time as a function of the number of Information Providers working concurrently.

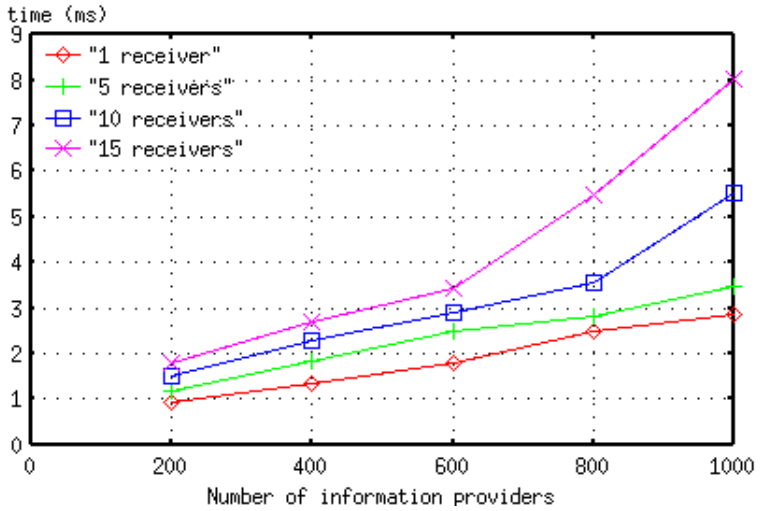


Figure 10-17 Time spent to transport one information object from one Information Provider to a number of Information Consumers vs. the number of concurrent Information Providers.

10.6 Integration tests

Online integration tests, binary integrations of the Online Software with other components, and experience from testbeam deployment will be presented.

10.7 References

10-1 Online Software Requirements
http://atlas-onlsw.web.cern.ch/atlas-onlsw/documents/documents_page.htm

10-2 Online Software Architecture
http://atlas-onlsw.web.cern.ch/atlas-onlsw/documents/documents_page.htm

10-3 Test Report of Large Scale and Performance tests, January 2003, in preparation

10-4 Summary document used as input to the ATLAS TPR document
Altas DAQ-1 technical notes
Conference Papers
http://atlas-onlsw.web.cern.ch/atlas-onlsw/documents/documents_page.htm

10-5 Notes on technology evaluation - to be written

10-6 References to external documents on used or evaluated technology

11 DCS

We consider that the level of detail presented in the chapter, as well as its structure should be respected. However, there we expect changes in the wording and in the logical connections between the different sections.

11.1 Introduction

The principle task of DCS is to enable the coherent and safe operation of the ATLAS detector. Safety aspects are treated by DCS only at the least severe level. All actions initiated by the operator and all errors, warnings and alarms concerning the hardware of the detector are handled by the DCS. Concerning the operation of the experiment, an intense interaction with the DAQ system is of prime importance.

+ Description of what is contained in this chapter and what it has been presented in chapters 1, 2, 3 and 5 (These chapters are needed before).

11.2 Organization of the DCS

The architecture of the DCS and the technologies used for its implementation are strongly constrained by environmental and functional reasons. The DCS consists of a distributed Back-End (BE) system running on PCs, which will be implemented with a Supervisory Control And Data Acquisition system (SCADA), and of the different Front-End (FE) systems.

The DCS FE instrumentation consists of a wide variety of equipment, from simple front-end elements like sensors and actuators, up to complex computer systems that are connected to the SCADA stations by means of standard fieldbuses. A SCADA run-time database contains records of all equipment where the data values are stored.

The equipment of the DCS will be geographically distributed in three areas as shown in figure XXX: the main control room at the surface of the installations, the underground electronics rooms USA15 and the detector's cavern, UX15. The SCADA component will be distributed over the two first locations while the front-end equipment will be placed in USA15, US15 and UX15 as shown in figure XXX.

The Front-End (FE) electronics in UX15 (see figure 1.3) is exposed to radiation and to a strong magnetic field up to 1.5 T. The instrumentation in the cavern must be radiation-hard or tolerant to levels of 1–10^5 Gy per year in the muon subdetector and inner tracker, respectively. In the following, only the DCS FE equipment located outside of the calorimeters in ATLAS where the dose rate is of the order of 1 Gy / year will be addressed.

The equipment in the detector's cavern will be interfaced via fieldbuses, to FE computer equipment located in the underground electronics room USA15 and US15. The former is accessible during operation and whereas the latter, will not be accessible during operation due to the remaining radiation levels. The equipment at this level will consist of workstations for subdetector experts for the supervision of individual partitions, mainly during commissioning and maintenance periods. It is foreseen to use a dedicated control computer for each detector. These systems are called Subdetector Control Stations (SCS) in figure *XXX Figure needs to be changed*

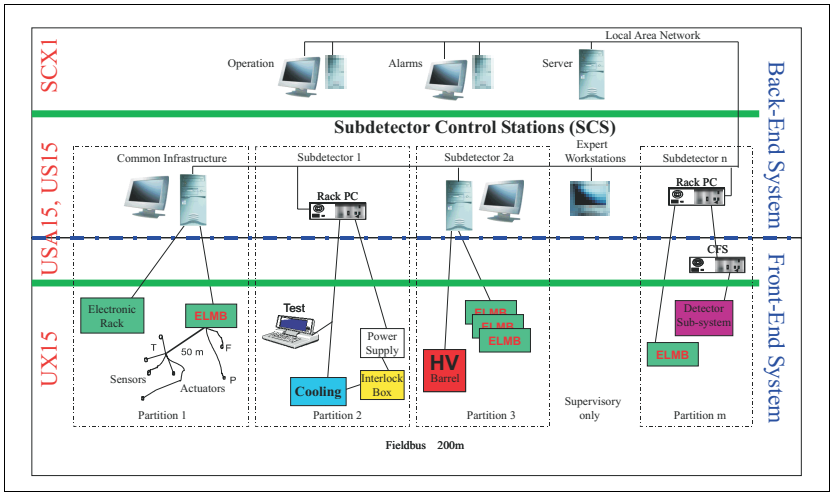


Figure 11-1 Geographical deployment of the DCS.

XXX and they will run the SCADA software collecting data from the front-end devices in their partition.

The Complex Front-end Systems (CFS) at this layer will not run SCADA and will be dedicated to specific tasks. The CFS are normally connected to SCS over a dedicated Local Area Network (LAN). CFS can also be placed in UX15 if they support the hostile environment.

The main control room will be located in the SCX1 building at the surface of the installations. This area will always be accessible to the personnel. The equipment will consists of general-purpose workstations, which will be linked to the control layer through a LAN providing TCP / IP communication. The workstations will retrieve information from the SCS of the different subsystems underneath and can be used to interact with them by means of commands or messages.

The DCS can be partitioned into vertical slices as shown in figure XXX (ref to the picture below). Such a partition can be operated completely independent from other slices of the DCS and offers full SCADA functionality to its users. A vertical slice controls a subsystems of the ATLAS detector, where a subsystem is defined as an arbitrary part of the detector, e.g. the high voltage system of a subdetector or the subdetector itself.

11.3 Front-End System

The front-end equipment consists of controllers, which connect to the hardware, either as separate modules or as microprocessors incorporated in the front-end electronics. Field instrumentation like sensors and actuators will be of various types and it will be tributary to the requirements for the detector hardware.

This equipment is distributed over the whole volume of the detector with cable distances up to 200 m. The distribution underground is governed by two conflicting constraints. Because of the

radiation level, magnetic field, limited space available for equipment and the inaccessibility at UX15 during beam time, the equipment should be located in USA15. However, complexity, cost and technical difficulties of cabling suggest condensing the data in UX15 and transferring only the results to USA15.

The harsh environment limits the types of technologies that may be used. CERN recommends a small number of fieldbuses and ATLAS has chosen the CAN fieldbus as the main standard for this area. The CAN fieldbus is reliable, may be used over large distributed areas, does not use magnetic sensitive components and has good support from industry. To ensure successful operation of equipment in the cavern, the “ALTAS Policy on Radiation Tolerant Devices” [ref] has been formed to give specific rules concerning testing and qualification of radiation tolerant electronics.

11.3.1 Embedded Local Monitor Board (ELMB)

Due to the harsh environment, the large number of channels required and the low cost required per channel, there is no commercial solution which exists. Therefore, a general purpose IO device has been developed called the ELMB.

The ELMB is a single, credit card sized PCB which may be embedded onto custom front-end equipment or may be used in a stand-alone mode. It has been designed with low power consumption so that it may be powered remotely via the fieldbus. It provides 64 high-precision analog input channels each of 16-bit accuracy. As well as the analog inputs, there are 8 digital input lines, 8 digital output lines and 8 configurable (either input or output) digital lines. Other interfaces are available such as a serial port allowing JTAG or other protocols to be implemented.

The standard software that is pre-loaded into the ELMB allows for communication over a CAN field bus using the higher level protocol CANopen. The standard functionality gives ‘plug and play’ usage for the analog inputs and digital inputs and outputs.

The ELMB fulfills the majority of requirements of the ATLAS subdetector applications in terms of accuracy, stability and functionality. It has been tested and qualified to operate in radiation and a magnetic field giving long term operation without manual intervention.

11.3.2 Other FE equipment

All four experiments in the LHC have similar requirements for front-end equipment. JCOP provides solutions and support for standard devices such as high and low voltage power supplies, PLCs and other common front-end equipment, as well as their interfaces into the SCADA system. ATLAS will use many of these standards as recommended for the four LHC experiments.

Other non-standard FE equipment has to be mentioned here: e.g. alignment and calibration systems.

11.4 The Back-End System

The functionality of the BE system is two-fold: It acquires the data from the front-end equipment and it offers supervisory control functions, such as data processing, displaying, storing and archiving. This enables the handling of commands, messages and alarms.

The BE system will be hierarchically organized to map the natural partitioning of the experiment into subdetectors, systems and subsystem. The BE hierarchy allows for the dynamic splitting of the experiment into independent partitions, which can be operated in stand-alone or integrated modes. The operation of the different subdetectors will be performed by means of Finite State Machines (FSM), which will handle the states and transitions of the different parts of the DCS. It is envisaged to have a FSM per subdetector. The coordination of the different partitions will be performed by means of commands and messages. The command flow is downwards, whereas the message exchange take place in either direction the within the slice. No horizontal communication is foreseen between different slices or amongst the components of an slice.

11.4.1 Functional Hierarchy

In order to provide the required functionality the BE of the DCS will be logically organized in three levels as shown in figure XXX. The actions on the operator time-scale are performed at the upper level, while the RT operations are performed at the lower levels. In addition, data and alarm archiving and logging of incidents and commands will be provided at each of these levels. Remote access to a well-defined set of actions to be performed by the two upper levels of the BE hierarchy will also be provided subject to proper access authorization.

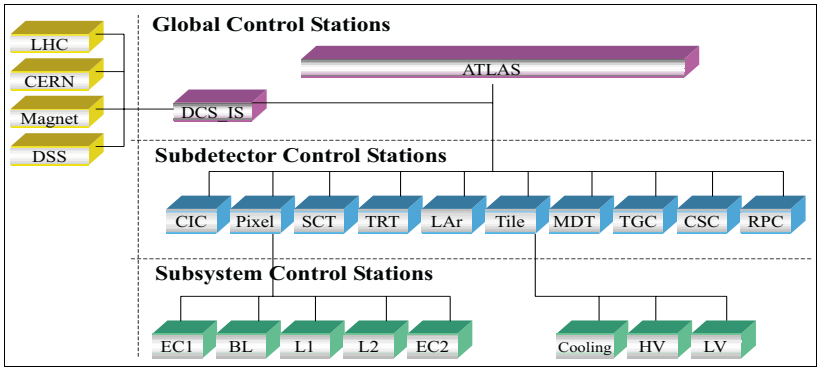


Figure 11-2 Hierarchical organization of the Back-End system of the DCS in three functional layers

Global Control Stations

The overall control of the detector will be performed by the uppermost level of the BE system, which consists of the Global Control Stations. These stations are envisaged to provided high level monitoring and control of all subdetectors and technical infrastructure. The full control of the detector is provided at only lower levels in the hierarchy. At this level, different services will be provided like the DCS Information Service to handle the communication with the external systems, namely the LHC accelerator, the CERN infrastructure and the Detector Safety System, or web and database services. Information for these subsystems will be used to build the overall status of the experiment. Bidirectional data exchange between the DCS and the TDAQ system will also be managed at this level. No command exchange between the TDAQ and the DCS is foreseen at this level.

Subdetector Control Stations

The Subdetector Control Stations are placed at the intermediate of the BE hierarchy. There will be one SCS per subdetector and an additional SCS to handle the monitoring of the common infrastructure in ATLAS called Common Infrastructure Controls (CIC). The later will be interfaced with the DCS Information service in the layer above. All actions on a given subdetector provided at the Global Control Stations are also provided at this level. In addition, the SCSs allow for the full and stand-alone local operation of the subdetector by means of dedicated graphical interfaces. The SCSs also handle the communication with the services of the layer above. It is foreseen to have a direct connection from the SCSs to the DCS Information service to provide the different SCSs with the status of the external system, namely the LHC accelerator, the Detector Safety system, CERN services and the ATLAS magnet, as well as with the environmental parameters of the common infrastructure. The SCS handle the co-ordination of all subsystems underlying in the layer below and are the responsible for the validation commands issued by the operator from the global control stations in the layer above or directly from the TDAQ run control. If low level control is required by the issued actions, e.g. ramp up high voltage, these commands can be propagated to the subsystems in the layer below for their execution. The overall status of the subdetector is assembled (collated???) and pass on to the TDAQ system via the DAQ-DCS communication software, which is described in section XXX, and to the control stations in the layer above.

Subsystem Control Stations

The bottommost level of the BE hierarchy is constituted by the Subsystem Control Stations, which handle the low level monitoring and control of the different systems and services of the detector. The organization of this level for a given detector could be performed attending to either geographical or functional criteria. In the former the arrangement follows the natural partitioning of the detector in sections, subsection, etc. whereas in the second approach, the organization is decided as a function of the different services of the subdetectors, e.g. cooling, high-voltage, etc. This level of the hierarchy is directly interfaced to the FE system. Besides the readout and control of the equipment, it also performs calculations and fine calibration of the raw data from the FE and comparison of the values with preconfigured thresholds for the alarm handling. The station placed at this level will executed the commands propagated from the SCSs in the layer above although they can also execute predefined automatic actions if required.

11.4.2 SCADA

The BE system of the ATLAS DCS will be implemented using a Supervisory Control And Data Acquisition (SCADA) product. SCADA systems [37] are commercial software packages normally used for the supervision of industrial installations. They gather information from the hardware, process the data and present them to the operator. Even though SCADA products are not tailored to LHC experiment applications, many of them have a flexible and distributed architecture and, because of their openness, are able to fulfil the demanding requirements of the ATLAS DCS.

Besides basic functionality like the Human Machine Interface (HMI), alarm handling, archiving, trending or access control, SCADA products also provide a set of interfaces to hardware, e.g. CERN recommended fieldbuses and PLCs, and software, e.g. Application Program Interface

(API) to communicate with external applications, or connectivity to external databases via the Open or Java DataBase Connectivity (OBDC and JDBC respectively) protocols.

SCADA products constitute a standard framework to develop the applications leading to a homogeneous DCS. Its usage saves development effort reducing the work for the subdetector teams. In addition, they follow the evolution of the market, protecting against changes of technology like operating system or processor platforms.

11.4.3 PVSS

A major evaluation exercise of SCADA products [38] was performed at CERN in the frame of the Joint COntrols Project (JCOP), which concluded with the selection of the PVSS-II, from the austrian company ETM, to be used for the implementation of the BE systems of the four LHC experiments.

PVSS is a device-oriented product where process variables that logically belong together are combined in hierarchically structured data-points. Device-oriented products adopt many properties from object-oriented programming languages like inheritance and instantiation. These features facilitate the partitioning and scalability of the application.

PVSS provides the interfaces to connect to external databases or systems, like the DAQ system or LHC accelerator, and the capability to extend the functionality of the product to interface custom applications or equipment (e.g. availability of driver development toolkit).

It is conceived as distributed systems. The single tasks are performed by special program modules called managers. The communication among them takes place according to the client-server principle, using the TCP/IP protocol. The internal communication mechanism of the product is entirely event-driven. This characteristic makes PVSS specially appropriate for detector control since, systems which poll data values and status at fixed intervals, present too big an overhead and have too long reaction times resulting in lack of performance.

The managers can be distributed over different PC running either Microsoft Windows or Linux. The communication between them is internally handled by PVSS-II. This has been one of the crucial points in the selection of this product since the DAQ system of the ATLAS experiment is been developed entirely under Linux, where as the DCS will widely use Windows.

PVSS allows to split the supervisory software into small application communicating over the network and it is imposed by the distribution of the DCS equipment in different locations in ATLAS.

11.4.4 PVSS Framework

Although PVSS-II will be used as the basis of the LHC experiment controls, this has been found not to be sufficient to develop an homogeneous and coherent system. An engineering framework on top of PVSS-II is being developed in the frame of JCOP. *XXXHas JCOP been mentioned before?XXX* The PVSS framework is composed of a set of guidelines, tools and components commonly used by the four LHC experiments like HV and LV systems. This framework will lead to a significant reduction of the development and maintenance work to be performed by the subdetector teams and to an homogeneous system by means of the standardization of the equipment utilized. It also addresses the interoperability of the different components included in the framework. The ELMB has been integrated into PVSS as a component of JCOP frame-

work with the aim of facilitating the usability of the ELMB node to the ATLAS users but also to ensure the homogeneity of the SCADA software. The ELMB component provides all PVSS infrastructure needed to work with the ELMB in a standard mode. This package also comprises a so-called a “top-down” configuration tool, which handles the configuration of all non-SCADA interfaces to the FE system.

All DCS systems from the global level to the local control stations will need some commonly used services. These applications will be implemented once and may be used at all levels. The main services to be provided will be for data and alarms, where both numerical displays and trending (with the native widgets and external trending tools) will be used, web based presentation of information and logging of all actions whether performed by an operator or an automatic process.

11.5 Integration FE-BE

The PVSS-II product will be used as SCADA system for the implementation of the supervisor layer of the ATLAS/DCS. There are several interfaces which allows to connect PVSS-II based systems to hardware.

- Dedicated drivers for PVSS-II; PVSS-II has the drivers for modbus devices, PROFIBUS and some others. It also contains the API to develop drivers by users.
- PVSS-OPC client; OPC is a wide used industrial standard. All commercial Low and High voltage systems are supplied with the OPC servers.
- DIM software, which is a communication system for distributed and multi-platform environments. DIM provides a network transparent inter-process communication layer developed at CERN.

The OPC due to the wide spread usage and the big support from industrial has been chosen as main interface from the SCADA to hardware devices. The main purpose of this standard is to provide the standard mechanism for communicating to numerous data sources. The OPC is based on the Microsoft Windows technology. The specification of this standard describes the OPC Objects and their interfaces implemented by OPC server. The architecture and specification of the interface was designed to facilitate clients interfacing to remote server. An OPC client can connect to more then one OPC Server, in turn an OPC Server can serve several OPC clients. All OPC objects, consequently, are accessed through interfaces.

In turn the ELMB is a CANopen device and will be widely used in the implementation of the subdetector front-end system.

CANopen is a high level protocol for the CAN-bus communication. This protocol is widespread as well. CANopen standardizes the types of CAN-bus messages (objects) and defines the sense of them. It allows to use the same software in order to manage of CAN nodes of different types and from different manufacturers.

To connect the ELMB to SCADA the OPC CANopen server has been develop. Others possibilities will also be used in suitable cases.

11.5.1 OPC CANopen server

On the market there are a lot of the CANopen servers. But all of them are tailored to their specific hardware interface cards and they do not provide the CANopen functionality required by the ELMB.

The OPC CANopen server works as the CANopen master of the bus handling network management task, node configuration and transmitting data to the OPC client. The OPC CANopen Server consists of two main parts.

- The main part is an OPC server itself. It implements all the OPC interfaces and main loops. Any application interacts with this part through interfaces. The CANopen OPC server transmit data to a client only on change, which results in a substantial reduction of data traffic.
- The second part “Can Bus component” is hardware dependent. It interacts with a CAN bus driver and controls CANopen devices.

Several busses with up to 127 nodes each, in accordance with CANopen protocol, can be operated by the OPC CANopen server. The system topology in terms networks and nodes per bus is modelled in start up time in the address space of the OPC CANopen server from a configuration file.

11.6 Read-out chain

The complete read-out chain ranges from the I/O point (sensor or actuator) to the operator interface and is composed of the elements described above: ELMB, CANopen OPC Server and PVSS-II. Apart from the data transfer, it also comprises tools to manage the configuration and the settings and status of the bus. PVSS-II models the system topology in term of CANbus, ELMB and sensor in the internal database by data-points. These data-points are connected to the corresponding items in the OPC server in order to send the appropriate CANopen message to the bus. In turn, when an ELMB sends a CANopen to the bus, the OPC server will decode it, set the respective item in its address space, which transmits the information to a data-point in PVSS. The different elements of read-out chain can perform the following functions.

The ELMB digitizes the analogue inputs and drives the digital input and output lines. Different settings can be applied to the ADC, including choosing as data output format either raw counts or calibrated micro-Volts. Data are sent either on request from the supervisor, or automatically in predefined intervals, or when they have changed. As the ELMB is in most cases exposed to ionizing radiation, it checks also for possible radiation-induced errors (e.g. memory or register changes) and tries to correct them.

The OPC server transmits data together with quality information when they have changed. It can optionally perform calculations, e.g. to convert the data into physical quantities in appropriate units.

The SCADA system applies the individual calibration procedures and compares the data with pre-defined thresholds. In this way warnings, alarms, and automatic actions are established. The SCADA also archives the data and allows its visualization.

11.6.1 Performance of the DCS readout chain

To investigate the performance and the scalability of the DCS readout chain to the size required by ATLAS, a full vertical slice (or full branch) consisting of 6 CANbuses having 32 ELMBs each was assembled.

The aim of this test was to study the behavior of the system with these characteristics to discover settings required in order to achieve the optimal results and to establish limits of the readout chain. These limits define the granularity of the system in terms of number of ELMB per CANbus and the number of buses per PVSS system. In particular, the following was to be inspected and investigated:

- Remote powering of the ELMB nodes via the bus. The radiation levels in the detector cavern impose that the power supplies will have to be placed in the underground electronics rooms US15 and USA15. Therefore, the power for the nodes will have to be fed remotely via the CANbus with distances up to 150 m.
- Bus loading, which determines the number of nodes per bus. The data traffic on the bus has to be uniformly distributed over time in order to keep the bus load low under normal operation. In ATLAS the bus occupancy will be kept below 60% in order to cope with a higher loads which may arise in case of problems like power cuts. In these cases, an avalanche of channel information which must be handled by the system.
- Optimization of the work balance amongst the different processing elements in the readout chain. The functions to be performed by the ELMB, CANopen OPC server and PVSS are homogeneously distributed to ensure equal load of each of these components and to avoid bottle-necks.
- Optimization of the system performance by tuning of different software settings such as update rates for OPC and the readout rate.
- Determination of the overall performance of the systems, which defines the number of CANbuses with these characteristics per PVSS system, and that will strongly condition the topology of the different subsystems.

The setup employed in the test, shown in figure 1. The system of CANbuses was operated from PVSS-II using the CANopen OPC server and a Kvaser CAN interface. The bus lengths were 350 m in all cases, in order to fulfil the ATLAS requirements with a broad margin. Up to 32 ELMB were connected at the end of each CANbus. The total number of channels in this system was: 12288 analog inputs, 3072 digital outputs, and 1536 digital inputs. It is important to note that the amount of channels in the set up described here, is of the order of magnitude of some large applications in ATLAS.During the test, the readout rate was increase in order to push the system to the limit. When big bursts of data arrive at PVSS-II very rapidly, the different messages are internally buffered. Two different situations can be distinguished:

- *Steady run*, where all messages sent by the ELMBs to the bus are stored to the PVSS-II database and, in addition, the CPU memory usage remains constant, i.e. no buffering is performed at the PVSS-II or OPC level.
- The so-called *avalanche run*, where the fastest possible read-out rate is estimated for a a short period, typically a few minutes. Under these circumstances, although all messages are archived to the PVSS database, the bus data flow is so high, that messages cannot be treated in real time and are buffered at the SCADA level therefore, leading to an increase of the memory usage. It is important to note that although long term operation under these conditions would not be possible, this situation can occur in case of major problems of the equipment monitored, like power cuts, and must be handled by the system.

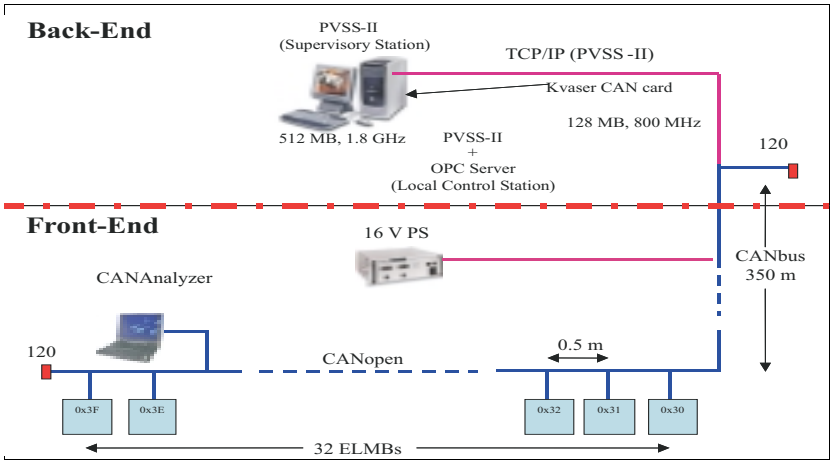


Figure 11-3 ELMB full branch test setup. *(To be changed: Show bus multiplicity)*

A readout rate of at least 30 s is required for a steady run in a system of 6 buses with 32 ELMB nodes each (12288 analog input channels). Under these conditions, the fastest readout rate in avalanche mode is limited to 8 s for several minutes. The examination of the CPU load showed that in all cases, the results presented are only limited by the CPU work load due to the PVSS-II managers as a consequence of the buffering of the CAN messages. These results indicate that the operation of the readout is strongly constrained by the performance of PVSS-II.

11.6.2 Long term operation of the readout chain

Mention as introduction that the DCS has to run 24/365.

The long-term operation of the full readout chain was tested with a number of ELMBs in a radiation environment similar to that expected in the ATLAS cavern, though with a much greater dose rate. This environment allows the different error recovery procedures implemented at the different levels of the readout chain, which are required due to radiation effects to be tested. A CAN bus of greater than 100 m was connected to a PC running the CANopen OPC server and PVSS-II. The test ran for more than two months, which was equivalent to more than 300 years at the expected ATLAS dose rate in terms of TID. The CAN controller in the ELMB ensures that messages are sent correctly to the bus, and will take any necessary action if errors are detected. Bit flips were seen during the test at the ELMB using special test software, and these are also handled at the ELMB level. The OPC server ensures all ELMBs on a bus are kept in the operational state, monitoring the messages on the bus in case of power glitches. At the highest level, PVSS scripts were utilized to monitor the current consumption for the bus (where increase in current is an indication of latch-up or damage from long term TID) and to reset ELMBs if communication has been lost. Through this script, the power supply for the bus was controlled allowing for hard resets to be performed. No user intervention was necessary during the time of the test.

11.7 Applications

The components and tools described above are used to build the applications, which control and monitor the experiment equipment. The applications for the supervision of the subdetectors is the responsibility of the subdetector groups and is described in the relevant TDRs. All equipment, which does not belong directly to a subdetector will be supervised by a SCS called Common Infrastructure Controls (CIC), which is hierarchically situated at the level of a subdetector. It monitors the subsystems described below.

All racks, both in the electronics rooms and in the cavern will comprise a control unit based on the ELMB. It monitors the basic operational parameters like temperatures, air flow, cooling parameters, etc. and also user-defined parameters. Some racks will have the option of controlling the electric power distribution. The crates which are housed in these racks have however usually their own controls interface.

General environmental parameters like temperature, humidity, pressure, radiation level etc. will also be monitored by the CIC. Parameters of the primary cooling system belong also this category. The individual subdetector cooling distribution systems are however supervised by the corresponding subdetector SCS.

All information collected is available to all other PVSS stations via a central DCS information server. A subset of them will also be transmitted to the DAQ.

11.8 Connection to DAQ

In order to grant a coherent functioning of both DCS and the physics data acquisition the following functionality of communication between that systems is to be provided [11-6, 11-7]:

- Bi-directional exchange of data like parameters and status values;
- Transmission of DCS messages, like alarms and other error messages, to DAQ;
- Synchronization DCS with TDAQ run control and providing ability for DAQ to issue commands on DCS (with feedback).

In accordance of the concept of TDAQ partitioning [11-3] the communication functionality required should be provided for each needing it TDAQ partition independently of others.

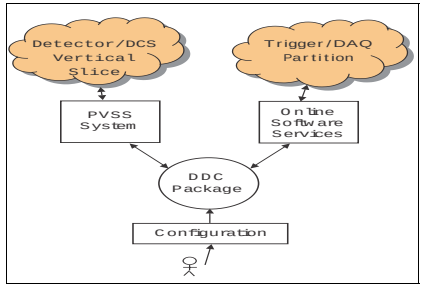


Figure 11-4 DDC package in relation with the DCS and the Trigger/DAQ system.

The TDAQ Online software package (see Ch.10) provides a series of services for Trigger / DAQ inter-application communications of the content declared above for DCS communication. They are the Information Service (IS) allowing to share the run time information (10.3.3.1), Error Reporting Service (ERS) distributing application messages (10.3.3.2) and the Run Control package (see 13.2) running a finite state machine to represent and control / synchronize the states of TDAQ subsystems of a partition. These services / subsystems will be used as the DAQ-side connection points for the communication with DCS.

The PVSS II product (11.4.2) is provided with a powerful application program interface (API) allowing full direct network access to the PVSS application runtime database. That API is DCS side law level interface for DAQ – DCS communication.

The DAQ – DCS communication package (DDC) is to be developed on top of the interfaces mentioned above as a generic tool configurable by end-user in terms of DAQ and DCS functionality (i.e., identification of data, messages and commands to be transferred). It provides the co-operation of DCS (PVSS) world and Trigger / DAQ world as it is illustrated in fig. 11.8.1

The underlying subsection describe the DAQ – DCS communication software components with their interfaces. The features belonging to all of them:

- Implemented as a PVSS API manager [11-8] integrates the program interface of corresponding Online software service;
- Waits if a communication partner is not running and recovers lost connection;
- Being configured from the TDAQ configuration database – this interface is omitted in figures below.

The prototype of the DDC package has been tried in the test beam experiments of 2002 – 2003 and demonstrated satisfactory and reliable capability of working.

11.8.1 Data Transfer Facility (DDC-DT)

The data exchange in both directions is to be implemented via the Information Service of DAQ Online software. The application keeps the data elements (parameters of the systems) declared in the DDC-DT configuration being the same in both source and destination of that data. This is done on the base of the subscription mechanism available for both sides. The possibility for DAQ of requesting single read of specified DCS data is also provided.. Figure 11.8.2 shows the interfaces to be used.

11.8.2 Message Transfer Facility (DDC-MT)

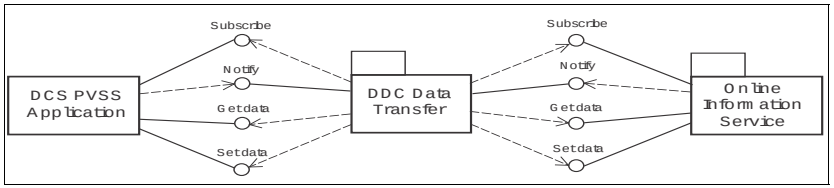


Figure 11-5 DDC data transfer interfaces.

The DCS message transferring to DAQ is to be implemented via the Error Reporting System of DAQ. The interfaces used are drawn in fig.11.8.3.

This component delivers for DAQ DCS alarm messages on appearance and DCS text variables to be interpreted as messages for DAQ, which have been defined in its configuration.

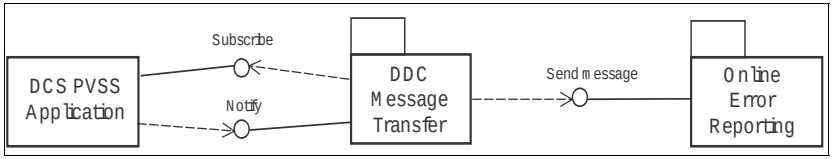


Figure 11-6 DDC message transfer interfaces.

11.8.3 Command Transfer Facility (DDC-CT)

The DDC-CT subsystem (being like other components a PVSS API manager) is implemented as a dedicated run controller (RC) to be included as a leaf into a TDAQ partition run control tree (see 13.2). This run controller, like any other TDAQ run controller, is capable to execute standard commands causing its transitions as defined by the TDAQ partition finite state machine. Except of that, it allows sending for DCS so-called *non-transition* commands (***nt-commands***) via the Online software information service. The DDC-CT interfaces are shown in fig. 11.8.4.

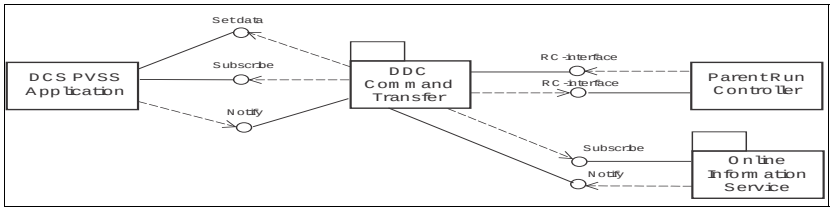


Figure 11-7 DDC command transfer interfaces.

The term *non-transition* emphasizes that those commands do not cause any finite state machine transition. An *nt-command* may be issued by any TDAQ application (including the parent run controller).

The content of a command (both run control and nt-command) and its execution is the responsibility of DCS's PVSS application. Mapping of the run control transitions onto certain set of commands on DCS is to be done in the DDC-CT configuration.

11.9 Interface to External Systems

The term External Systems designates systems having their own control system with which the DCS has to interact. ATLAS will gain access to external systems via the DCS. We can distinguish two main external systems: the CERN technical infrastructure and the LHC accelerator. The former consists of a number of subsystems like cooling and ventilation, electricity distribution, radiation monitoring, etc. The Detector Safety System (DSS) and the Magnet Control System are also considered part of the technical infrastructure. All these external systems must be concurrent into the general DCS. Although these systems are designed to react in case of problems, early indications of their status must be notified to the DCS since they may have consequences onto the detector and automatic corrective actions, driven by the DCS, may be required. The DCS will reflect also the states of all these systems and, in many cases, will act as their user interface.

The connection will support bidirectional information exchange and, in some cases, the sending and receiving of commands. This interface will be unique for the 4 LHC experiments and it will be developed in the framework of the JCOP.

11.9.1 CERN Technical Services

The technical services around the ATLAS detector include cryogenics and conventional cooling, ventilation, gas system, electricity, radiation monitoring, low and high voltage power supplies. The DCS will also have access to the control and status of infrastructure including AC mains, air conditioning etc. These services will monitor the environment to guarantee the safety of the personnel and equipment and will enable the different subdetectors of the experiment to function within their required operating conditions. Some of the systems will need feedback from the subdetector to operate. This is the case for the gas system and cooling, where part of their equipment consist of external stand-alone PLC or commercial I/O modules, whereas some information come from the detectors themselves. Therefore, slow closed-loops maybe needed between the DCS and this type of system.

11.9.2 Detector Safety System

As previously mentioned, the DCS is not responsible for the security of the personal nor for the ultimate safety of the equipment. The former is the responsibility of the LHC-wide hazard detection systems, whereas the latter has to be guaranteed by hardware interlocks and stand-alone PLC and is the responsibility of the Detector Safety System. Although the information exchange between the DCS and the DSS must be bi-directional, actions must go only in one direction. The DCS must not disturb the operation of the safety system. However, warnings about problems detected by the safety system must be notified to the DCS in order to take corrective actions or to shut down the problematic part of the detector. Control access will also be handled by the CERN services and it will be needed at the DCS side.

11.9.3 Magnet system

Due to its critical requirements [10] and complication, a dedicated PLC-based control system will be implemented for the ATLAS magnet. The operator will not need direct control, although a detailed online status and knowledge of all important parameters of the magnets is essential for the operation of the detector and for the subsequent physics analysis. This dedicated system will supervise and control the cryogenics, the cooling system, the power supplies and the instrumentation of the magnet.

11.9.4 LHC

An robust interface between the experiment and the accelerator must be provided. Instantaneous beam parameters like the different types of background, beam position and luminosities observed in the detector must be transferred from the experiment to the accelerator for consequent tuning of the beam.

The experiment will also give all information on its status such as status of its magnets, in particular, the solenoid which acts directly on the beams, status of sensitive equipment like high voltage on the sub-detectors and other status signals as well as global status signals such as the operation state of the detector, setting up, etc. The DCS has to make sure that the detector is in an appropriate state (e.g. voltage settings) before LHC is allowed to inject particles.

ATLAS may need the possibility to request actions like a fast beam dump should the backgrounds become dangerous for the subdetectors or injection inhibit.

On the other hand, machine parameters like status signals for setting up, shut-down, controlled access, stable beams, beam cleaning must be transferred from the accelerator to the experiment. The machine should also provide information on the beam like emittance, focusing parameters, energy, number of particles and a horizontal and vertical profile needed for offline physics analysis. Information on the vacuum conditions in the vicinity of and in the experimental straight section, and position of the collimator are also of interest to the experiment.

The LHC has dedicated instrumentation for the comprehensive measurement of all these parameters. The subset of operational parameters of the accelerator, relevant to the operation of the detector or to the subsequent physics analyses have to be delivered to the DCS and must be logged.

Although the exchange of many of these parameters is only needed during data-taking, a subset of this information, like the integrated radiation doses in the different parts of the detector measured by the DCS, has to be known to the LHC at all times. Therefore, this communication is required regardless the state of ATLAS. This is one main reason why this communication will be handled by the DCS on the ATLAS side and not by the DAQ system.

11.10 References

11-1 *New references to be added*

11-2 H.J. Burckhart et al., “Vertical Slice of the ATLAS Detector Control System”, submitted to 7th Workshop on Electronics for LHC Experiments, September 2001, Stockholm (Sweden).

11-3 F. Varela Rodriguez et al., “ELMB Full Branch Test: Behaviour and Performance”, ATLAS DCS Internal Working Note 13, October 2001.

11-4 F. Varela Rodriguez. “The Detector Control System of the ATLAS experiment: An application to the calibration of the modules of the Tile Hadron Calorimeter”, PhD. Thesis, CERN-THESIS-2002-035, April 2002.

11-5 http://www.kvaser.com

11-6 V. Filimonov, “Description of the CANopen OPC server v2.5”, http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/DCS/ELMB/DOC/OPCCOUserGuide.pdf

11-7 ***References from Connection to DAQ:***

11-8 H.Burckhart, M.Caprini, R.Jones “Connection DCS - DAQ in ATLAS”, ATLAS DCS IWN8, Nov 1999,

11-9 http://atlasinfo.cern.ch/ATLAS/GROUPS/DAQTRIG/DCS/dcs_daq_0.6.pdf.

11-10 R.Hart, V.Khomoutnikov “ATLAS DAQ - DCS Communication Software. User Requirements Document”, Nov 2000,

11-11 http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/DCS/DDC/ddc_urd.pdf

11-12 <TDAQ Partitioning document> Probably, made already earlier at the document

11-13 <Reference to PVSS> Probably, made already earlier at the document

11-14

12 Experiment Control

12.1 Introduction

The Experiment Control involves the supervision and coordination of the operational state of the detector and its parameters and the control of the instrumentation and software involved in the event readout An architectural overview of the Experiment Control has been introduced in chapter [ref. to Architecture chapter]. The specific architecture of the systems involved has already been discussed in the previous chapters. Now the interaction of the various systems and building blocks which provide the general control functionality required by the experiment is described. Scenarios for physics data-taking and for calibration modes are discussed.

As already presented in the architectural view, TDAQ control and DCS control are two complementary and interacting systems. These systems have different tasks and requirements. Whilst the TDAQ control is only required when taking physics or calibration data and during detector commissioning and tests, the DCS has to operate with no interruption to ensure the safety of the detector. The DCS is based on a SCADA system [12-5], while the TDAQ control is based on the TDAQ Online Software [Chapter 10]. The necessary synchronisation to control systems which are external to TDAQ is provided via DCS.

12.2 Control Coordination

The control of the experiment is given by the interplay between three systems: The LHC machine, the Detector control and the TDAQ control. For each of the them the status of the system under control is expressed in distinct states.

12.2.1 Operation of the LHC machine

The phases of the LHC define a multitude of states [12-2] important for the internal functioning of the machine. A subset is of direct interest for the interaction with the experiment control, in particular those states which describe the condition of the beam with consequences for the operation of the detector. Phases with stable beam and collisions indicate that the detector is operational for data-taking.

The main phases of interest here are of the following type: *filling* the beam from the SPS into the LHC, *ramp*, when the beam is accelerated up to its nominal energy, *squeezing* the beam, prepare for physics and *collide*, *physics* with stable beam, beam *dump and ramp-down* and *recover.*

The various phases will be indicated directly by the LHC operation and by observation of LHC equipment.

12.2.2 Operation of the DCS as a State Machine

The DCS must enable the stand-alone operation of the sub-detectors, as well as the coherent and integrated operation of all sub-detectors for concurrent Physics data-taking. For these reasons,

the operation of the different sub-detectors will be performed by means of Finite State Machines (FSM). The FSM approach allows for sequencing and automation of operations and it supports different types of operators and ownership, as well as, the different partitioning modes of the detector required to fulfil the control needs in the various scenarios presented in the following. The FSM will handle the transition of the different parts of the DCS through internal states. It is envisaged to have one FSM per sub-detector and an additional FSM for the global operation of the experiment. The states of the sub-detector FSM will be assembled from the status of the different parts or services of the detector, which are determined by the status of the FE equipment, and from the status of the various environmental parameters monitored by the CIC station. The states of the external system, interfaced via the DCS_IS described in section 11.8.1, will also be considered, e.g. the state of the LHC accelerator.

The global operation of the BE system will be performed by a single FSM whose states will be built from the states of the different sub-detectors' FSM, previously configured, and the status of the external systems. Any transition issued at this level will be propagated to the underlying sub-detectors' FSM included in the running mode of the experiment.

12.2.3 Operation of the TDAQ States

Three main TDAQ states from *initial* to *stand-by* and *running* have been introduced in Chapter 3.1. Here the states are further sub-divided as explained in [12-3] as shown in Figure 12-1. Two states are placed between *initial* and *running.* Before arriving to the *initial* state the software infrastructure is initialized. The loading of the software and configuration data is performed which brings the system to the *loaded* state. The system configures the involved hardware and software and enters the *configured* state and the TDAQ system is ready to start data-taking. In the subsequent *running* state the TDAQ system is taking data from the detector. Data-taking can be paused and the L1 busy is set.

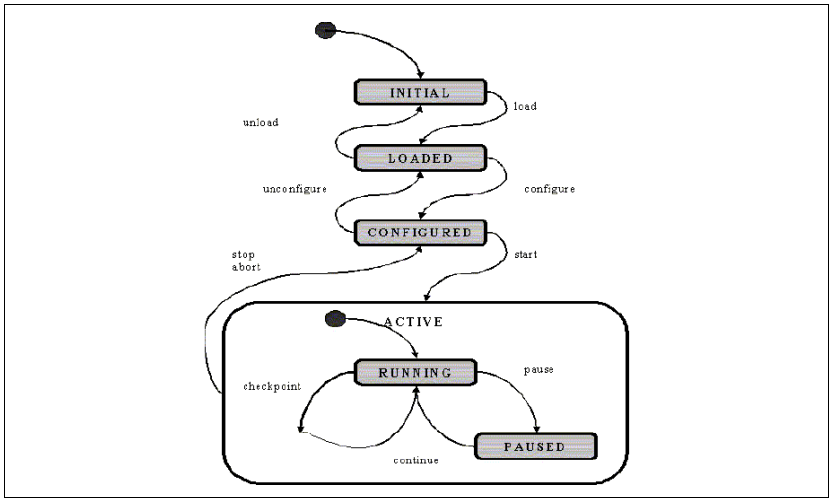


Figure 12-1 TDAQ states

The checkpoint is a transition in a running TDAQ system which is triggered by a change in conditions or by an operator. It results in the following events to be tagged with a new run number and does not need the synchronisation, via run control start / stop commands, of all TDAQ elements. Some components in the TDAQ control system require TDAQ sub-states which will be used for synchronisation during certain transitions.

12.2.4 Connections between States

As it has been presented in the previous sections, the LHC, the DCS and the DAQ system will each be operated through states. The synchronization of these systems is required in order to ensure coherent data taking and the integrity of the detector. The communication with the LHC will be handled by means of the DCS as described in Chapter 11. The DCS monitors the status of the LHC continuously and transfers this information in real time to the DAQ system in order to prepare the detector for Physics data-taking. On the other hand, parameters measured by the TDAQ system like beam position or individual bunch luminosity, can be used to tune the beams and therefore, must be transferred to the LHC via the DCS.

The actions on the sub-detector hardware performed by the DCS will have to be coordinated with the states of the LHC machine to ensure the safe operation of the sub-detectors. This is the case of the ramping up of the high voltage of some sub-detectors, like the Pixel or SCT trackers. These sub-detectors are more vulnerable in case of insufficiently focused beam if the high voltage is on. For these sub-detectors this kind of actions on the detector will only be taken if the accelerator provides stable beams. For this reason, the DCS states will closely follow the operation of the LHC operation. The LHC state will be related to a pre-defined set of operational conditions of the sub-detectors DCS and of the DAQ system. Periods of particle injection or acceleration in the LHC may be used by the DCS or the DAQ to initialize and configure the different parts of the systems, like the front-end electronics. Information on the internal states of the DCS will be transferred to the DAQ system via the DDC described in section 11.8 . Once the safe operation of the experiment instrumentation is ensured from the LHC machine, the DCS will bring the sub-detectors to the required state for data-taking and it will communicate their availability to the DAQ system by means of the DDC. During Physics data-taking there will be an intense bi-directional communication between these systems. *An example is the request by the DCS to the LHC for beam dump if large backgrounds are observed by the DAQ system in the detector.*

The TDAQ control is only partially coupled to the LHC and the DCS states depending on the type of run. For physics runs it must be ensured that the LHC is providing stable beam and collisions are taking place and that DCS has brought the detectors into the corresponding state. The TDAQ system can generally be brought from *initial* to the *configured* state while the LHC is ramping, sweezing and preparing for physics, and while the DCS prepares the detector for data-taking. Then is ready for taking physics data and waiting in a stand-by mode for LHC and DCS to be ready. For some of the calibration runs similar conditions apply for the selected detector. For other calibration, for example with cosmic rays or with an external source the co-ordination with the DCS is required but no co-ordination with the LHC states. For most TDAQ system tests no c-ordination with other states need to take place.

may-be add a diagram

12.3 Sub-system Control

The control functionality of the building blocks presented in the architectural view will be described in the following paragraphs.

12.3.1 Online Software Control Concepts

remark: here the terms detector and sub-detector are used, this can be changed to whatever terms for those items will be decided to be used for the TDR.

The TDAQ system is given by a large number of hardware and software components, which have to operate in a coordinated fashion to provide for the data-taking functionality of the overall system. The organisation of the ATLAS TDAQ system in detectors and sub-detectors leads to a hierarchical organisation of the control system. The basis of the TDAQ control is provided by the ATLAS Online Software, which is explained in detail in chapter 10.3.

The basic element for the control and supervision is a controller. The TDAQ control system is build of a large number of controllers which are distributed in a hierarchical tree following the functional composition of the ATLAS TDAQ detector.

The concept is illustrated in Figure 12-2. As illustrated, four principle levels of control are currently foreseen. Additional levels can be added at any point in the hierarchy if needed. A top level controller named root controller has the overall control over the TDAQ system. It supervises the next level of controllers in the hierarchy, the detector controllers or sub-system controllers. It is the responsibility of the detector controller to supervise the hardware and software components which belong to this detector. The next control level takes the responsibility for the supervision of the sub detectors which correspond to the TTC partitions [12-4]. On the lowest level the so-called local controllers are responsible for the control of Read-out crates and alike. Farm supervision and ROS hardware make use of the same controllers following a similar structure, which is further discussed in 12.3.2 Data Flow Control and 12.3.3 HLT Farm Supervision.

A controller in the TDAQ system is characterised by its state following the TDAQ state model described in Section 12.2.3. In any place of the hierarchy, a change of state is initiated and synchronized from the higher level controller and sent down to the next lower level. From there information is returned to the next higher level when the requested transition has been performed. Possible error conditions are also reported back to the next higher level.

A controller framework allows to handle the operations described above in a coherent way on all the controllers in the system. On the other hand, it also gives the necessary flexibility to the detector expert to customize each controller for handling the individual tasks on the system under its control. These tasks take a wide range of variety from read-out hardware to event filter farm control. The exact information on the relationship of the controllers and their responsibilities is contained in the configuration database as detailed in the chapter 10.4.3 Architecture of Databases.

The controllers have a number of responsibilities: Each controller is responsible for the initialisation and the shutdown of software and hardware components in its domain. It is also responsible for passing commands to child controllers and for signalling its overall state to its parent. Of particular importance is the synchronisation necessary to start the data-taking. This is performed by successive transitions through a number of intermediate states until data-taking is fi-

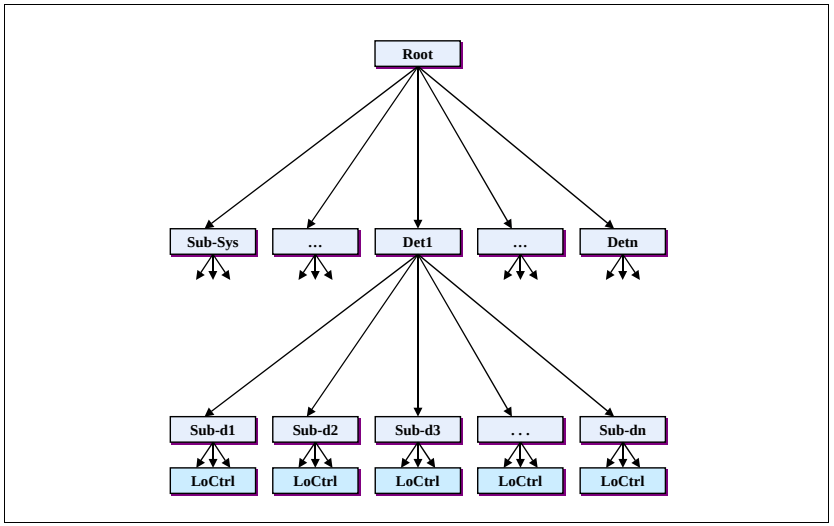


Figure 12-2 Online Software Control Hierarchy in TDAQ

nally started as described below in 12.4.1 Initialisation, Data-taking and Shutdown Phase. Interaction with the shift operator via the user interface drives the operations via commands to the highest level controller. The inverse series of synchronized transitions is traversed when data-taking is stopped. If necessary, it is envisaged to introduce so called hidden states to allow for further synchronisation points on the sub-system level.

During all the operational phases, each controller is responsible for the supervision of the operation of elements under its direct control and for the observation of the operations of its children thus providing also for the task of error handling. In case of a malfunction of a detector, the controller can start corrective actions and/or signal the malfunction by sending messages. Severe malfunctions which are beyond the capabilities of a controller can be signalled by a state change to its parent. It is then the role of the parent controller to take further actions. The design of the control, supervision and event handling functionality is based on the adoption of a common expert system shell. Specific nodes will use different rules to perform their functions in addition to a common rule base which handles the generally valid aspects.

12.3.2 Data Flow Control

The Dataflow control encompasses the ROS/ROD control and the Data Collection control.

The Dataflow control is comprised of the control of all applications and hardware modules responsible for moving the event data from the detector front-end electronics and LVL1 trigger to the high level triggers (LVL2 and EF). It includes the control of the ROD crates, the RoI Builder, the ReadOut System and the Data Collection applications, such as the Event Builder.

There are two flavours of local controllers in the DataFlow foreseen, both making use of the Online software infrastructure in the same way. The ROS controller is tailored towards the control

of ROS software applications and hardware devices which cannot themselves access the online software facilities and the DC controller which handles the different types of DC applications and is optimized for the control of computer farms. A version of the latter is also used for the control and supervision of the high level triggers and is further described in Chapter 12.3.3. The main difference between the two controllers is that the one which controls ROD crates and the Read Out System cannot assume that it controls active and intelligent elements. A ROD is a hardware device on which no standard software application is running: in this case the controller is the only access point to the databases as well as the only element communicating over IS/MRS to the Online system.

Both controllers can be deployed at different levels of the control hierarchy. As an example, a Data Collection controller can be used as top controller for all event building applications, as well as controller for a group of them. In general, such a controller can be in charge of other sub controllers or of endpoint data taking applications, transparently.

The Data Flow controllers make use of the configuration database to extract the information on the elements they are supposed to supervise. Their duty is to start, control and stop the data taking elements (hardware and software), to monitor the correct functioning of the system, gather operational statistics information and perform local error handling for those kinds of errors which couldn't be handled by the data taking nodes, but do not need to be propagated further to higher control levels.

12.3.3 HLT Farm Supervision

The emphasis for HLT control is on the management of the Computer farms. It is assumed that the farm for a HLT is divided into a set of subfarms, each under control of a specific controller. These controllers have well defined tasks in the control for the underlying processing tasks.

The High Level Triggers (HLT) perform the final selection before sending events to permanent storage. They consist of the 2nd Level Trigger (LVL2) and the Event Filter (EF). The two stages of the HLT are implemented on processor farms, divided into a number of subfarms. A key design principle has been to make the boundary between LVL2 and EF as flexible as possible in order to allow the system to be adapted easily to changes in the running environment (luminosity, background conditions, etc.) Therefore commonalities between the two sub-systems need to be exploited as fully as possible. Bearing this in mind, a joint control and supervision system has been envisaged.

The Online Software configuration database will describe the HLT in terms of the software processes and hardware (processing nodes) of which it is comprised. The HLT supervision and control system will use the configuration database to determine which processes need to be started on which hardware and subsequently monitored and controlled. It is foreseen that the smallest set of HLT elements which can be configured and controlled independently from the rest of the TDAQ system (i.e. a "segment") will be the subfarm. This allows subfarms to be dynamically included/excluded from partitions during data-taking without stopping the "run". Supervision and control for each subfarm will be provided as a local run controller, which will interface to the Online Software run control via a farm controller. The controller will provide process management and monitoring facilities within the subfarm. The controller will maintain the sub-farm in the best achievable state by taking appropriate actions, e.g. restarting crashed processes.

Where possible, errors should be handled internally within the HLT processes. Only when they cannot be handled internally should errors be sent to the supervision and control system for further consideration.

It is foreseen that Online Software services will be used by the supervision system for monitoring purposes. For example, IS will be used to store state and statistical information which could be displayed (for example) by a dedicated panel in the Online Software graphical user interface.

12.3.4 Detector control

In order to provided all the functionality required for Physics and calibration runs, the DCS system must provide the flexibility to model the partitioning schema of the DAQ system. The finest granularity of the DAQ system, is given by segmentation of the sub-detectors in TTC zones. For these reasons, the different sections of the sub-detectors will be logically represented in the BE software of the DCS by means of the so-called control units, which will be operated as FSM. According to this model, the DCS of the Tilecal, for example, may be organized in four independent control units, which would control the four sub-detector sections. The control units will be hierarchically organized in a tree-like structure to reproduce the organization of the experiment in sub-detectors, DCS system and sub-systems as illustrated in Figure 12-3.

Each control unit may control a sub-tree below consisting of other control units or device units, which are responsible for the monitoring and control of the equipment.

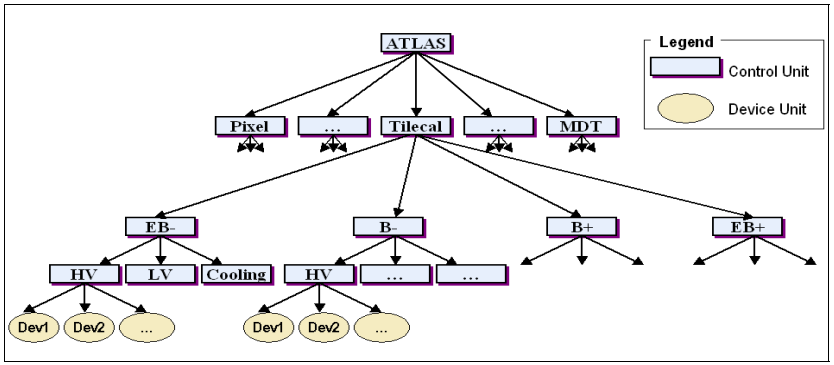


Figure 12-3 DCS Logical Architecture.

In order to map the dynamic structure of the DAQ system, the control units will support different partitioning modes. Any control unit may be excluded from the hierarchy and operate in stand-alone mode for testing, calibrations or debugging of part of the system. The different modes of operation of the DCS, which allow for stand-alone or integrated mode with the DAQ system, will require the implementation to handle the ownership of the different control units. This mechanism will be developed according to the recommendations of the JCOP Architecture Working Group [12-1].

12.4 Control Scenarios

12.4.1 Initialisation, Data-taking and Shutdown Phase

Pre-conditions, DCS state, state of external systems

The data-taking states as described in Chapter 12.2 are traversed when the system is run through initialisation, data-taking and shutdown phases. When initiating a data-taking session, operation starts from booted but idle machines. The Operator chooses the configuration to be used. The infrastructure consisting of a number of servers in the distributed system is started. Initialisation of the infrastructure hardware and software is performed and the correct functioning of the system is verified. Sequence and synchronisation of these start-up operations follows the dependencies described in the configuration database Section 10.4.1.1. The communication services are provided via the Online Software Information sharing services and allow for information and message exchange as well as for monitoring and histogramming. Once the infrastructure is in place, the controller processes and the application processes which are part of the configuration are booted. In the distributed system, process management and synchronisation is de-centralized and can therefore occur in parallel.

Once all processes have been booted successfully the operator can cycle the system through the states which are used to synchronize the configuring of hardware equipment and software application which take part in the data-taking process. During the loading transition, the initialisation of all the processing elements in the system including for example the loading of the software and configuration data is performed. During the following transition, called configuring, the configuration of a loaded system, for example the realisation of connections between TDAQ elements or the setting of parameters, is preformed.

When these operations are terminated, the system is in a state ready to take data. The operations described up to here may be time-consuming and can therefore be performed a significant time before starting the run, for example when waiting for stable run conditions.

The operator can now give the signal to start a run via the graphical user interface to the system. The L1 busy is removed and event data-taking operations are activated. If found necessary, a run can be paused and resumed later in an orderly manner. The transitions involved should only concern the direct data-taking activity to minimize time overhead. On the occurrence of special conditions [ref] the checkpoint transition as described in [ref] can lead to a change in run number.

It could be described here what can happen here while running,; parts of it is described in Chapter 12.4.2, "Control of a Physics Run" at the moment:

- *partition de-coupling*
- *partition joining*
- *sub-farm removal, sub-farm joining*
- *component failures and error handling [ref to chap.3 and 6]*

When the operator stops the run, all data-taking activities are stopped. The involved control and application processes remain active. (*describe actions in DF, HLT, L1 for unload - unconfigure).* On receipt of the shut-down command, clean-up operations are performed by the software application and in the hardware and the controllers and the previously started applications are

stopped. Then the infrastructure is removed in an orderly manner in order to leave the system in a state in which a new and independent data-taking session can be started.

12.4.2 Control of a Physics Run

remark: here the terms sub-detector and sub-sub-detector are used, this can be changed to whatever terms for those items will be decided to be used for the TDR.

The overall control system must ensure the safe and coherent operation of all sub-detectors in an integrated mode for concurrent physics data-taking. The online control system, which handles the control of all the read-out elements in the system and the DCS which controls the detector hardware, are synchronized by means of commands issued from the TDAQ system to the DCS. The interface between DAQ and DCS, called DDC, has been described in detail in 11.8 . It consists of three aspects. Dedicated communication interfaces for data exchange, presented in section 11.8.1 and for message exchange, described in section 11.8.2 are available. During all data-taking phases, bi-directional data and message exchange between both systems may take place from any level of the DCS BE to the Online software information sharing and message reporting services. The command communication between TDAQ and DCS is provided via a dedicated controller called DDC_CT, which has been introduced in section 11.8.3

The TDAQ will be the master system and will drive the data-taking. Figure 12-2 shows the organization of the Back-End (BE) system of the DCS and the TDAQ hierarchy of controllers presented in Section 12.3.1. For Physics data-taking, all TDAQ controllers will be integrated in a single common partition.

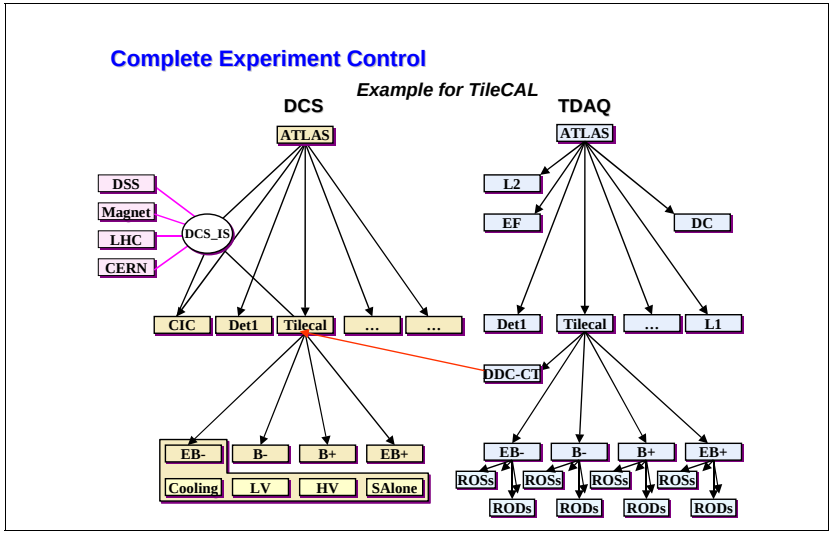


Figure 12-4 Complete Experiment Control mode. *Figure to be changed. DCS part should show a logical organization*

It is envisaged to have one DDC_CT controller per sub-detector. This controllers will send commands

directly to the Subdetector Control Stations (SCS) on the DCS side. No command flow is foreseen from the TDAQ system and the DCS global operation station. The DCS SCS are directly connected to the sub-detector services underneath as well as to the external systems by means of the DCS Information Server (DCS_IS). All information required to operate the detector is available at the SCS level. For these reasons, the SCS will have local decision capabilities. Therefore, the SCS represent the natural place to validate and execute commands issued by the TDAQ system, since they can cross-check the status of the different systems to ensure the safe operation of the detector.

This communication model implies that the TDAQ system will interact directly with the DCS FSM of the various sub-detectors. The availability of each of the partitions of the sub-detector for Physics data-taking will be notified to the TDAQ system from the SCS by means of the message transfer facility of the DDC.

Only a pre-defined set of high-level commands from the TDAQ system like the triggering of state transitions on the DCS side will be allowed. The interpretation and execution of the commands is the entire responsibility of the DCS, since the TDAQ system will not have the knowledge of the underlying structure of the DCS. The state of the command execution on the DCS side will be reported to the TDAQ system directly by means of the DDC_CT. The TDAQ Online software control system handles failures or time-outs from the DDC_CT in the same way as from other controllers in the system.

The TDAQ system control operates according to the hierarchical Online Software Control concept as further described in Section 12.3.1. As illustrated in Figure 12-4, the detector controllers, the farm control for EF and L2 as described in Section 12.3.3, and the DC controller operate at the same level in the hierarchy. Each detector controller supervises the sub-detector controllers and the controller which provides the detector connection to DCS. ROSs and RODs are supervised by the respective sub-detector controller. Global error handling and recovery is provided by the Online system control.

In case of malfunctioning of a detector or sub-detector, the respective partition can be removed from the control and read-out partition tree. The global partition control will continue to supervise the read-out if the detector part in question is not vital for data-taking for the type of physics chosen at the time. It can run in stand-alone mode to allow detector experts to repair eventual problems and join later the global control partitions and its read-out chain.

HLT sub-farms can be removed or added to the global farm control without disturbance of data-taking actions. Breakdown and replacements of individual sub-farm nodes will be handled transparently and each of such operations will be logged.

this may have to be expanded in more detail

12.4.3 Calibration Run

the different types of calibration runs will be described similar to the description of the physics run

- *Three different type of calibration runs: Pure TDAQ, for example test pulses for LA calibration, pure DCS (calibration of the temperature sensors, adjustment of the cooling flow depending on the temperature) or both systems are involved, for instance the case of the calibration of the Tile Calorimeter using the Cs source.*
- *Shall we refer to only the third case in this section? My personal impression is yes.*

- *In all calibration where both systems are required, the TDAQ system will be the master of the process.*
- *The picture shows the case in which the Tilecal detector is operated in stand-alone mode for calibration where both systems are involved.*
- *The DCS will execute the commands issued from the TDAQ by means of the DDC_CT.*
- *case stand-alone of sub-sub-detector, also more than one in parallel*

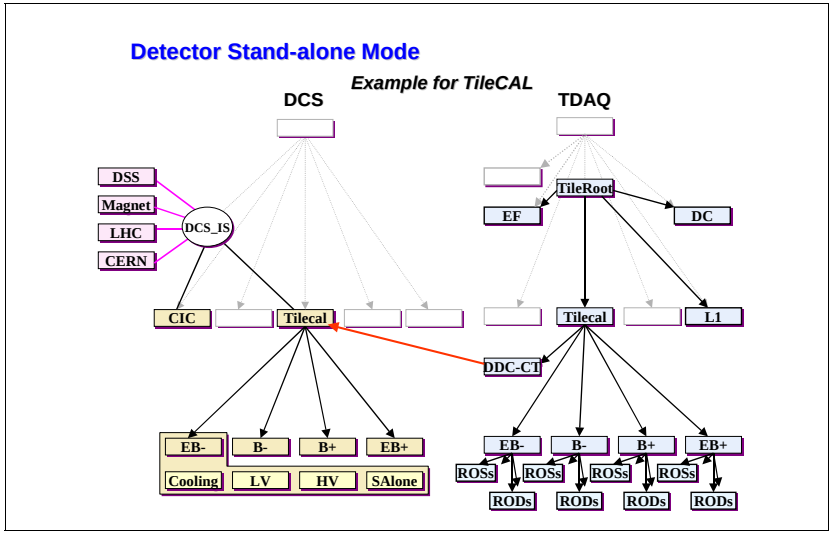


Figure 12-5 Detector Stand-alone mode. *Figure to be changed. DCS part should show a logical organization*

12.4.4 Operation outside a Run

During shut-down periods and long term intervals without data-taking, the full functionality of the DCS will be required in order to supervise the operation of the sub-detectors and common services. In this situation, the DCS becomes the master of the detector. In order to reduce the cost, the operation high power consumption equipment or the circulation of expensive gases in the sub-detectors will be interrupted in these periods. However some sub-detector services like the LAr and ID cryogenics, will continue to be operated during these periods. The monitoring and control of the humidity and temperature of the electronics racks, the supervision of the uninterruptible power supply system and of other sub-detector specific equipment will be required in order to enable a safe operation. For these reasons, the access of the DCS to the conditions and the configuration databases must also be ensured outside a run. During these periods, the DCS will also handle the communication with the external systems. The ATLAS magnet will be permanently switched on and therefore, the interface with the DCS must be continuously available. The radiation levels monitored by the LHC control system must be accessible by the

DCS at all times. The DSS will be able to trigger actions on the DCS in case of problems under these circumstances. Similarly, the interface to the fire brigade and to the access security system must be continuously operational.

In this scenario, the DCS will allow for the operation of the sub-detector in stand-alone mode, as required for debugging of the system or calibrations, and in an integrated mode. In the former case, the operation will be performed from the sub-detectors control stations having full control of the sub-detector, whereas in the latter, the overall control will be performed from the global operation station and only a subset of high level actions will be possible on the sub-detectors as the triggering of transitions between sub-detector states.

12.5 References

12-1 JCOOP Architecture Document

12-2 LHC Operations project: http://lhcop.web.cern.ch/lhcop/

12-3 Runs and States - Global issues working group document: http://atlas-project-tdaq-giwg.web.cern.ch/atlas-project-tdaq-giwg/Documents/Documents.htm

12-4 Partitioning - Global issues working group document: http://atlas-project-tdaq-giwg.web.cern.ch/atlas-project-tdaq-giwg/Documents/Documents.htm

12-5 The SCADA system...

Part 3

System Performance

13 Physics selection and HLT performance

13.1 Introduction

In the Technical Proposal for the HLT, DAQ and DCS of the ATLAS experiment, a first understanding of the on-line event selection scheme and the corresponding physics coverage was made. Since then, the studies have evolved, to cope with different machine scenarios and additional constraints coming from the detector itself. One of the major changes to take into account has been the LHC start-up phase, currently foreseen to deliver 10 fb^{-1} in one year, with a peak luminosity per fill of $L = 2.0 \times 10^{33}$ $cm^{-2}s^{-1}$, a factor of two with respect to the Technical Proposal assumptions. This change has motivated a complete revisiting of the approach to the Physics and Event Selection Architecture of the experiment, leading to a novel way of reducing event rates and sizes, while retaining as much as possible of the ATLAS physics goals. Needless to say, only the availability of real data will allow this proposal to find a concrete implementation and the tuning of the relative weights of the selection will only be possible then, when confronted with the environment of LHC data taking.

As it has been explained in Chapter 9, the High Level Trigger system of the experiment is composed of two separate data reduction steps, the Level-2 (LVL2) and the Event Filter (EF), each of them with distinctive and complementary features. The common denominator of these selections is that they will operate using software algorithms running on commercial computers to validate the hypotheses of particle identification. The LVL2 will do this with purpose built algorithms that need to operate in about 10 ms and use only part of the detector information at full granularity, the EF will have the fully built event at disposal, with a latency of the order of a second. An important aspect is to maintain a flexible scheme allowing for an easy adaptation to changes in conditions like luminosity or background: the modularity of the HLT will allow the implementation of different reduction steps at different stages.

Given these commonalities and these distinctions, it has been recognized that a coherent and organized approach to the software components of the trigger validation was needed to make a fundamental step forward with respect to the TP. The work that will be presented in Section 13.2 has concentrated on this issue, by deriving the common tools for the event selection and identifying the data model components and methods that can be shared across the different algorithms, in particular at LVL2. This will ease the implementation of different selection schemes, by making as well simpler the migration across levels.

Another important focus point for new developments has been the compliance with the updated detector geometry and with the realistic format of the data coming from the Read Out System. This implies that algorithms will operate on streams of bytes organized according to the read-out structure of each detector, in exactly the same way in which they will in the real experiment. This has allowed to study and understand the implication of converting those byte-streams to the objects needed by algorithms in order to perform trigger selections as well as making preliminary measurements of the overheads stemming from these conversions.

In Section 13.3 the outcome of present studies are presented. Particular emphasis has been put on the selection of electrons and photons, and on the one of muons. For those “vertical slices” of event selections, a thorough implementation of the approach described above has been attempted. After LVL1 validation, data organized according to the read-out format are used by LVL2 algorithms, operating within the framework of the PESA Steering and Control (refPESA). Trigger elements are then built using detector information and verified against hypotheses of particle identification. If LVL2 validation is successful, the EF reconstruction and analysis is performed (seeded or not by the LVL2 result) and the final selection published for off-line use. Rejection against dominant backgrounds and efficiencies for typical signals are reported, as well as the rates deriving from each of the selections.

To fully span the ATLAS physics coverage, also signatures involving jets, taus, ETmiss, as well as jets with b-quark content have been studied, and results are reported in the same section. As described in Chapter 4, the available on-line resources will also be used, for luminosities below the peak one, to evaluate *b*-production cross-section and make precision measurements with *B*-hadrons.

The global assessment, based on the present evaluation for each signature, of the ATLAS rates to off-line is made in Section 13.4, together with a preliminary description on how to reduce further the data volume by applying compression techniques or zero suppression to the detector information. A sketch of issues related to the initial phase of the experiment seen from the selection architecture point of view is also given in Section 13.5.

13.2 Common tools for selection

The basic software components which provide data to derive the trigger decision are the HLT algorithms. These algorithms operate within the context and environment of the PESA Core Software which is discussed from a conceptual design and architectural standpoint in ***Chapter 9*** provides an overview and description from the viewpoint of HLT algorithms. HLT algorithms manipulate and exchange data via objects of a common Event Data Model described in Section [Ref: sect:edm]. Furthermore, algorithms rely as much as possible upon common tool components which are discussed in Section [Ref: sect:tools]. An inventory of HLT algorithms intended to operate in the LVL2 environment is given in Section [Ref: sect:lvl2]; the inventory for EF algorithms is given in Section [Ref: sect:ef].

13.2.1 Algorithmic View of the Core Software Framework

HLT algorithms must allow themselves to be guided by the PESA Steering, to be seeded by Trigger Elements, and to operate with a restricted set of event data.

The Trigger processing starts from a LVL1 RoI using predefined Sequences of algorithms. These LVL1 RoI objects are decorated by Trigger Elements to allow them to be acted upon by the Steering. For each of these Trigger Elements, the Steering executes the required algorithms as defined in a Sequence Table. Hence, it is possible that a given algorithm may be executed N times per event. This is fundamentally different than the "Event Loop" approach of the Offline reconstruction paradigm where a given Offline algorithm would act only once upon each event.

To accomplish the Steering of algorithms using Sequence Tables and Trigger Elements, a Seeded approach is required. Trigger Elements characterizing abstract physics objects have a label (e.g., "electrons" or "jets") and effectively decouple the Steering and Physics Selection from details of the Event Data Model used by the algorithms. Via the Navigation scheme within the PESA Core Software environment, algorithms may obtain concrete event data associated with a given Trigger Element which define the Seed of restricted and relevant event data fragments upon which they should work. Again, this is different than the data access paradigm of the Offline environment where a given Offline algorithm would have potentially full access to event data.

At LVL2, event data reside within ROBs until actively requested. This allows the LVL2 algorithms to request and process only a small fraction of event data from ROBs, representing a substantial reduction in the network and computation resources required. The first step in this process is the conversion of a geo-

metrical region (*e.g.*, a cone with an extent η and ϕ) into Identifiers; this is accomplished with the HLT RegionSelector.

The HLT RegionSelector translates geometrical regions within the fiducial volume of the detector into a set of Identifiers. Presently these Identifiers are `IdentifierHash`es corresponding to elements of appropriate granularity in each sub-detector, usually a DetectorElement. As such, the RegionSelector uses DetectorDescription information during its initialization phase to build an `EtaPhiMap` for each layer (or disk) of a subdetector. This map is essentially a two-dimensional matrix in η and ϕ. Each element consists of a `list` of `IdentifierHash`; the column indices are ϕ floating point numbers while a range (η_{min}, η_{max}) specifies row indices. The input to RegionSelector API is the sub-detector under consideration (*i.e.*, Pixel, SCT, TRT, LAr, Tile, MDT, RPC, CSC, or TGC) and the extent of the geometrical region. Given the vastly different designs of each subdetector, a subdetector-dependent procedure is used. With knowledge of the layers and/or disks in the region, the RegionSelector searches the $\phi \rightarrow$`IdentifierHash` map which will give a set of `IdentifierHash` is relevant in ϕ region. The last step is to validate each `IdentifierHash` inside the `IdentifierHash`$\rightarrow (\eta_{min}, \eta_{max})$ map.

Interactions with the Data Collection system are hidden from the Algorithm behind a call to StoreGate. Within StoreGate, event data are aggregated into collections within an IdentifiableContainer (IDC) and labelled with an Identifier. Algorithms request event data from StoreGate using the set of Identifiers obtained by the HLT RegionSelector. If the collections are already within StoreGate, it returns them to the HLT algorithm. If not, StoreGate uses the IOpaqueAddress to determine which ROBs hold the relevant event data and requests it from the Data Collection system. A ByteStream converter converts the Raw Data into either Raw Data Objects (RDOs) or, by invoking a DataPreparation AlgTool, into Recontruction Input Objects (RIOs). The obtained RDOs or RIOs are stored within the collections within the IDC within StoreGate.

13.2.2 Event Data Model Components

During 2002 and 2003, there has been a substantial ongoing effort within the HLT, Offline, and subdetector communities. The goal of this effort has involved the establishment of a common EDM between HLT and Offline software in the areas of the raw and reconstruction data models. In the discussion that follows, the concept of a DetectorElement is used as an organizing and identifying principle for event data model objects; these are discussed in Section [Ref: sect:DE]. Currently, there has been convergence with respect to the raw data model described in Section [Ref: sect:rawedm]. Common reconstruction data model classes specific to LVL2 and EF algorithms have been developed and are described in Sections [Ref: sect:recedm] and [Ref: sect:ropo].

13.2.2.1 Event Data Organization

Event Data (*e.g.*, Raw Data Objects (RDOs) and Reconstruction Input Objects (RIOs)) are aggregated into collections corresponding to adjacent readout channels within the physical detector. These collections reside in an IdentifiableContainer (IDC) with Identifier labels corresponding to the unit of aggregation. For most sub-detectors, the organizing principle is that of the DetectorElement.

In the Pixel detector a DetectorElement is a module, equivalent to a single Silicon wafer; hence there are 1744 Pixel DetectorElements. For the SCT, a DetectorElement is one side of a module, equivalent to a bonded pair of wafers whose strips are oriented in a single direction (*i.e.*, axial or stereo); there are 8176 SCT DetectorElements. For the TRT, a DetectorElement is a planar set of straw tubes representing one row at a given radius of straws in a barrel module (*i.e.*, a plane corresponding to the tangential direction in the barrel) and 1/32 in $r\phi$ at a given z of straws in an end-cap wheel (ref:idedm) there are 19008 TRT DetectorElements.

For the calorimeters, the concept of DetectorElement does not exist (but it should). Instead, the organizing principle for event data is that of the Trigger Tower.

Within the muon spectrometer, for the MDTs, a DetectorElement is a single MDT chamber, where there is at most a single MDT chamber per station, and typically, an MDT chamber has two multilayers. An RPC DetectorElement is the RPC components associated to exactly one barrel muon station; there may be 0, 1 or 2 RPC doublet sets per station and a doublet set may comprise 1, 2 or 4 RPC doublets. A TGC DetectorElement is one TGC eta division, or chamber, in a TGC station; there are 24 forward stations in a ring and 48 endcap stations in a ring and there are four rings at each end of the ATLAS detector. Finally, for a CSC DetectorElement is a single CSC chamber, where there is at most a single CSC chamber per station. And typically, a CSC chamber has two multilayers.

13.2.2.2 Raw Data Model Components

This subsection summarizes aspects of ByteStream Raw Data, Raw Data Objects (RDOs), and Converters between the two for each sub-detector.

ByteStream Raw Data is Read-Out Buffer (ROB)-formatted data produced by the ATLAS detector or its simulation. It is defined by a set of hierarchical fragments, where only the bottom level, the ROD fragment, is defined by the sub-detector group. The format of the ByteStream has not yet been formally defined. Hence, preliminary “best guesses” have been made as to its structure which may undergo changes in the future.

A Raw Data Object (RDO) is uncalibrated Raw Data converted into an object representing a set of readout channels. Historically this has been referred to as a *Digit*. It is the representation of Raw Data which is put into the Transient Event Store (TES) and could potentially be made persistent.

The purpose of the RDO converters is dual: first a Raw Data ByteStream file can be created by taking the information from the already filled RDOs (in the transient store, from ZEBRA); second, this ByteStream file can then be read back by the converters to fill the RDOs (or the RIOs for LVL2). Since the RDOs are a representation of the specific detector output, its content can change with the life time of the sub-detectors.

13.2.2.2.1 Inner Detector

The content of the ROD fragment for the Pixel, SCT, and TRT detectors is given in Table [Ref: tab:InDetROD].

The implementation of the RDOs for the Inner Detector is based on a design reflective of the possibility that the content of RDOs may evolve in time. Hence there is the possibility of having several concrete classes of RDOs. The access to the content of the word that constitutes the raw information is done through member function that apply the right bit field masks to the word. Finally, the implementation of the RDOs makes use of the IdentifiableContainer (IDC) base class for a quick access to the stored collections, which, for the Inner Detector, has a granularity corresponding to DetectorElements in each sub-detector.

[Descriptive text about Pixel Raw Data goes here. Need input from Marina Cobal, Lorenzo Santi, et al.]

[Descriptive text about SCT Raw Data goes here. Need input from Jochen Schieck et al.]

The TRT readout and resulting Raw Data format is complicated due to its dual purpose as a tracking and particle identification detector and the nature of reading out a wire/straw detector. The TRT ByteStream consists of data collected in a 75ns window following the start of and LHC bunch-crossing which generates a LVL1 accept. Hence, the readout period is three times the nominal 25ns *time-slice* separation between LHC beam crossings. Two species of readout data exist: a tracking output using a 200eV discriminator threshold (referred to as *Low Threshold*) and a particle identification output using a 5000eV discriminator threshold (referred to as *High Threshold*). In recording the low threshold signal, eight bits for each of three consecutive time-slices are used. Each 25ns time-slice is divided into eight 3.125ns wide bins. A bit is set to one if the low threshold discriminator was on during the corresponding time bin. This leads to 24 bits being used to record the low threshold data. In recording the high threshold signal one bit for each time-slice is used. The bit is set to one if the high threshold discriminator was on at any time during the time-slice. This leads to a total 3 bits being used to record the high threshold data. The resulting set of 27 bits is referred to as "full encoding" and contains data for three time-slices with the earliest data occupying the highest order bits.

13.2.2.2.2 Calorimeters

The content of the LAr and Tile Calorimeter ROD fragments is given in Table [Ref: tab:rodcal].

For the LAr calorimeter, the current mapping is that all front-end boards (FEBs) in one Feedthrough is put into one ROD. The LAr RDO, `LArRawChannel`s, are stored in `LArRawChannelCollection`. These collections can be accessed through `LArRawChannelContainer`, a subclass of `IdentifiableContainer`. Currently, the `LArRawChannel`s in a `LArRawCollection` is ordered according to temporary definition of LAr Trigger Tower (TT) Identifier as defined in `LArIdentifier` package. This provides access to RoIs expressed as a group of TT. To access all `LArRawChannel`s in a RoI, the user can use the selector class, `LArTT_Selector` in `LArRawUtils` package, given a `LArRawChannelContainer`, and a vector of TT Identifiers.

Even though in the standard Offline data model `LArCell`s are stored in `CaloCellContainer` in granularity of subdetectors (*i.e.*, EM, HEC, FCAL), a special Collection/Container similar to that for `LArRawChannel`s is provided for LVL2 algorithms since it is desirable to create `LArCell` directly from ByteStream for efficiency reasons. The calibration of `LArCell`s, normally applied by algorithms in `LArCellRec` packages, have to be applied. This is achieved by defining a `LArCellCorrection` base class, a subclass of `AlgTool`, in `LArRecUtils` package, and the converter will retrieve a set of these `AlgTool`s from `ToolSvc` to apply the corrections to the `LArCell`s. This guarrantees the `LArCell`s created by the ByteStream converter is identical to the `LArCell`s created in the offline data processing sequence.

The granularity of the Tile Calorimeter from readout point of view is the following. There are 4 independent systems (ROS) which correspond to 2 halves of barrel and 2 extended barrels. Every such a cylinder contains 64 modules (or half-modules in case of barrel) and output from every 8 modules goes into one ROD unit. Therefore one ROD fragment contains 8 sub-fragments which correspond to 8 modules. Also, there is a possibility to have different kind of sub-fragments, for example sub-fragment with raw digits (25ns time slices) and sub-fragment with reconstructed quantities like energy, time, quality. There is no limit on total number of sub-fragments in one Tile ROD fragment: they are put one after another and identified by 32-bit words. Upper half of this word is fragment type and lower half is fragment ID The fragment ID itself can be split into 2 parts: 8 bits ROS number (range 1-4) and 8 bits module number (range 0-63). For the moment only one type of sub-fragment is implemented, namely sub-fragment which contains amplitude, time and quality. All these values are member elements of `TileRawChannel`. The amplitude in `TileRawChannel` is expressed in ADC counts (10 bits range), time is in nanoseconds and quality is the number in the range 0-1. Although it might not be necessary, all variables are treated and signed. The content of the this fragment is given in Table [Ref: tab:rodcal].

As is the case for the LAr Calorimeter, a special Collection/Container similar to that for `TileRawChannel`s is provided for LVL2 algorithms since it is desirable to create `TileCell` directly from ByteStream for efficiency reasons. The calibration of `TileCell`s, normally applied by `TileRawChannelToCell` algorithm in TileRecAlg package, have to be applied. A temporary solution for calibration is to hold all calibration constants in the `TileInfo` class. `TileInfo` is stored in TDS and then retrieved by any algorithm when needed. Both `TileRawChannelToCell` and `TileROD_Decoder` are using the same constants from `TileInfo` class This guarantees the `TileCell`s created by the ByteStream converter is identical to the `TileCell`s created in the offline data processing sequence.

13.2.2.2.3 Muon Spectrometer

The definition of RDOs for the RPC is complicated by the way the RPC read-out is organized. The RPCs are trigger chambers, and the organization of their read-out is oriented toward this task: the read-out structure does not reflect any easily identifiable geometrical structure, like for example, a single RPC chamber with its strips.

The primary task of the RPC trigger chambers is to record hits in the *pivot plane* (*i.e.*, the middle one of the three RPC planes) that coincide in time either with hits in the RPC plane closest to the Interaction Region ("confirm plane low p_T") or with hits in the RPC plane farthest from the Interaction Region ("confirm plane high p_T"). The RPC strips are read out via coincidence matrices (CMs). The η strips of the pivot plane and the η strips of the low p_T (high p_T) confirm plane are connected to the low p_T (high p_T) CMs and likewise for the ϕ strips. The CMs are grouped together in PADs. Each PAD reads out all coincidences between RPC planes in a 3-dimensional volume. Neighbouring PADs overlap in η on the two confirm planes but not on the pivot plane. There, a PAD covers half a RPC chamber. Per logical sector, there can be up to 8 PADs. The cabling of the RPC strips to the CMs is such that one CM reads out either the η or the ϕ strips in half a PAD. Consequently, one PAD reads out two low p_T η CMs, two low p_T ϕ CMs, two high p_T η CMs and two high p_T ϕ CMs.

The RPC ByteStream format reflects the read-out subdivision in PADs that are subdivided in CMs that read-out fired CM channels. Note that there is no simple one-to-one correspondence between RPC strips and CM channels. In order to determine which RPC strip fired given which channel fired in a, say, low p_T η CM, a cable map and the information in the high p_T ϕ CM of the same PAD is needed. The latter information is necessary to resolve the ambiguities caused by the fact that more than one RPC strip can be connected to the same CM read-out channel.

There will be three different types of RPC RDOs:

1. Bare RDOs;
2. RDOs with prepared data, first variety;
3. RDOs with prepared data, second variety ("digits").

The bare RDOs are the minimal object representation of the information contained in the bytestream. Three classes are used: `RpcPad`, `RpcCoinMatrix`, and `RpcFiredChannel`.

`RpcPad` is a `DataVector` of `RpcCoinMatrix` which, in turn, is a `DataVector` of `RpcFiredChannel`. `RpcFiredChannel` contains the online identifiers of the CM channels (not the RPC strips)

that fired. Objects of type `RpcPad` are the collections that are stored in an IDC. An object of type `RpcPad` is the smallest unit of information a HLT algorithm can retrieve from StoreGate with a single request. Accordingly, an object of type `RpcPad` is uniquely identified by means of an Offline Identifier as defined in Ref. cite{ref:rpc2}. These Offline Identifiers follow the geometrical RPC structure, not the read-out structure. Since the projections of the PADs onto the pivot plane do not overlap, the location where the PAD intercepts the pivot plane can be identified uniquely with an Offline Identifier and is used to identify an object of type `RpcPad`.

The class hierarchy in the first variety of the RDOs with prepared data is identical to the one in the bare RDOs. The only difference is that `RpcFiredChannel` contains the information which RPC strips fired (their Offline Identifier). To arrive at the RPC strip that fired access to the cable map (*i.e.*, to information external to the ByteStream), is needed. The HLT algorithm muFast will make use mainly of this type of RDOs.

The second variety of RDOs with prepared data corresponds to RPC digits and are used by EF algorithms like Moore.

[Need text for MDTs here, I will contact Monika G. and Ketevi.]

13.2.2.3 Reconstruction Data Model Components

RIOs are the most upstream objects that algorithms typically interact with (as opposed to RDOs or RodInputDigits).

13.2.2.3.1 Inner Detector

Similar to the RDOs, the implementation of the RIOs makes use of the `IdentifiableContainer` base class, and the collections are also according to the granularity of DetectorElements.

The Pixel and SCT RIOs are Clusters. A Cluster in the Pixel detector is a two-dimensional group of neighbouring readout channels in a DetectorElement. A Cluster in the SCT is a one-dimensional group of neighbouring readout channels in a Detector Element. For Pixel and SCT, there are currently two implementations of the Cluster class: one used for EF and Offline and one used for LVL2. The one used at EF has Pixel and SCT sharing the same class. For LVL2 there is a common structure for Pixel, SCT and TRT, but they all have their own concrete classes. For Pixel and SCT there is a base class used for LVL2. There is also an *Extended* class which could potentially be used at EF (which inherits from the LVL2 base class) in the future. Both LVL2 and EF set of cluster classes contain a list of RDO identifiers from which the cluster is built. The number of member functions is limited in both set of classes and the member functions follow the InnerDetector Requirements cite{ref:idedm}. It is assumed that in the future there will be only one set of RIO classes to be used for LVL2, EF, and Offline.

The TRT RIO is the drift circle of a straw. In the case of the TRT, the same classes are used for LVL2, EF, and Offline: those classes are the `DriftCircle` classes part of the set of classes that are also used at LVL2 for Pixel and SCT. The granularity of the TRT RIO is the same as for the RDO: that of a straw, thus the RIO contains an identifier which is the offline identifier for a straw. In the case of the RDO the straw information is uncalibrated and is just the direct content of the detector output, while in the case of the RIO the straw information is calibrated: out of the drift time, a drift radius is obtained. For now, the drift function applied is the same for all straws. In the future the constants that go into the parametrization of this drift function will come from the Interval of Validity Service cite{ref:IoVS.

13.2.2.3.2 Calorimeters

For the Calorimeters, the RIOs are calibrated calorimeter cells (`LArCell`s and `TileCell`s), imported from the offline reconstruction.

Both `LArCell`s and `TileCell`s have `CaloCell` as a common base class which represents the basic nature of a observation in the ATLAS calorimeters an energy, position, time and quality. A `CaloCell` has been calibrated so that `energy()` returns the physical energy deposit *in the cell* with units of GeV, but without any kind of leakage corrections. Time represents when the feature extraction thinks the deposit occurred, in nanoseconds, relative to the trigger. It ought to be zero for good hits. Quality reflects how well the input to the feature extraction system matched the signal model on which the feature extraction algorithm is based. It is a number, from zero to 1, giving the significance of the hypothesis that the actual signal is a sampling of the signal model (*i.e.*, it is the integral of a probability distribution from -\inf up to an observed value of a test statistic and ought to be uniformly distributed from [0,1] if the hypothesis is correct).

A UML class diagram of `LArCell` and `TileCell` is given in Figure [Ref: fig:caluml].

13.2.2.3.3 Muon Spectrometer

[Need text here.]

13.2.2.4 Reconstruction Output

13.2.2.4.1 Tracks

A track is, in general, an object containing a parametrization of a hypothesized particle trajectory through space relating groups of RIOs and/or SpacePoints together. A Track trajectory consists of three position, two direction, and one curvature [Footnote: The use of curvature assumes a homogenous magnetic field in which case this quantity is constant. For ATLAS and its significantly inhomogenous magnetic field in the endcap region of the Inner Detector and in the Muon Spectrometer, this parameter may be replaced by an invariant quantity such as *charge*/*p*.] parameters. If a track is evaluated at an intersecting surface, there are five parameters and a covariance matrix.

A proposed uniform Track class exists for LVL2 algorithms, the `TrigInDetTrack` class. A UML class diagram of `TrigInDetTrack` and associated classes is shown in Figure [Ref: fig:track]. No such uniform Track class yet exists in the Offline environment. [Footnote: There are of course Track classes defined internally within ORPs such as iPatRec and xKalman++.]

13.2.2.4.2 Calorimeter Clusters

[To be written]

13.2.3 Tools for HLT Algorithms

13.2.3.1 SpacePoint Formation

LVL2-specific clusters (*i.e.*, `SCT_Cluster` and `PixelCluster` give the position of the Clusters within the local 1 (2)-dimensional coordinate system of the SCT (Pixel) DetectorElement. These are converted to 3-dimensional coordinates in the ATLAS global coordinate system using the `AlgTools` `SCT_SpacePointTool` and `PixelSpacePointTool`. These tools accept as input a STL `vector` of pointers to Cluster Collections of the appropriate type, `SCT_ClusterCollection` or `PixelClusterCollection`, and return a STL `vector` of objects of the class `TrigSiSpacePoint`. A UML class diagram of the LVL2-specific SpacePoint class `TrigSiSpacePoint` and associated `InDetRecInput` classes is shown in Figure [Ref: fig:spacepoint]

For the Pixels, the creation of SpacePoints consists of combining the local coordinates of Clusters with information on the position and orientation of the DetectorElement to give the global coordinates.

The process for the SCT is more complicated since a single SCT detector provides only a one-dimensional measurement. However, an SCT module, consisting of two detectors in a stereo-pair, provide 2-dimensional information. One species of SCT DetectorElement, phi-layer, has strips orientated parallel to the beam axis, the other, *u* or *v* layer, is rotated by ± 40mRad with respect to the phi-layer DetectorElements. The formation of SpacePoints consists of the following steps:

- Associate each phi-layer Cluster Collection [Footnote: There is a Cluster Collection per DetectorElement.] with the corresponding stereo-layer Cluster Collection;
- For each pair of Collections (phi + stereo), take each phi-layer Cluster and search for associated stereo-layer Clusters. If there is more than one associated stereo layer Cluster, a SpacePoint is formed for each (in this case one, at most, will be a correct measurement, the others will form "ghost" points). If no associated stereo-layer hit is found, a point is created from the phi-layer information alone;
- Calculate the second coordinate (*z* for the barrel, or *R* for the end-caps);
- Using information on the position and orientation of the DetectorElement transform to global coordinates.

Note that for the LVL2 SpacePoints some simplifications are made in the interest of speed, as follows:

- No attempt is made to form SpacePoints from Tracks passing close to the edge of a module, where the corresponding stereo-layer Cluster is in a different module.
- Since the stereo and phi layers are separated by a small distance, the trajectory of the track will influence the measurement of the second coordinate. Since the trajectory is not known at the time that SpacePoints are created, there will be a corresponding increase in the uncertainty in the measurement in the second coordinate (*R* or *z*).

13.2.3.2 Track Extrapolation

[To be written]

13.2.4 HLT Algorithms for LVL2

13.2.4.1 IDSCAN

IDSCAN is a track reconstruction package for LVL2. It takes as input SpacePoints found in the Pixel and SCT Detectors. A series of sub-algorithms (`ZFinder`, `HitFilter`, `GroupCleaner`, `TrackFitter`) then processes these and outputs Tracks and the SpacePoints associated with them.

The `ZFinder` finds the *z*-position of the primary interaction vertex. The algorithm puts all hits into narrow ϕ-bins and extrapolates pairs of hits in each bin back to the beam-line, storing the *z* of intersection in a histogram. It takes as the *z*-position the histogram region with the most entries.

The `HitFilter` finds groups of hits compatible with Tracks from the *z* position found by `ZFinder`. It puts all hits into a histogram binned in ϕ and η. It then finds clusters of hits within this histogram. It creates a *group* of hits if such a cluster has hits in more than a given number of layers.

The group of hits found by `HitFilter` is used by `GroupCleaner` which splits groups into Tracks and removes noise hits from groups. Each triplet of hits forms a potential track for which p_T, $\phi 0$, and $d0$ are calculated. It forms groups from these triplets with similar parameters, applying certain quality cuts. It accepts a track candidate if a group contains enough hits.

Finally, the `TrackFitter` verifies track candidates and finds the track parameters by using a standard Kalman-filter-type fitting algorithm adapted from SCTKalman cite{SCTKalman}. It returns a list of SpacePoints on the Track, the Track parameters, and an error matrix.

13.2.4.2 SiTrack

SiTrack is a track reconstruction package for LVL2 which extends and upgrades a previous algorithm called PixTrig. SiTrack takes Pixel and SCT `SpacePoints` as input and outputs fitted reconstructed Tracks, each storing pointers to the `SpacePoints` used to build it. SiTrack is implemented as a single main algorithm `SiTrack` which instances and executes a user defined list of sub-algorithms (chosen among `STSpacePointSorting`, `STMuonVertex`, `STTrackSeeding`, and `STThreePointFit`).

`STSpacePointSorting` collects pointers to `SpacePoints` coming from the Pixel and SCT detectors and sorts them by module address, storing the result in a Standard Template Library (STL) map. This processing step is performed in order to speed-up data access for the other reconstruction sub-algorithms.

`STMuonVertex` is a primary vertex identification algorithm mostly suitable for low luminosity events with an high p_T muon signature. It is based on track reconstruction inside the LVL1 muon RoI: the most impulsive track is assumed to be the muon candidate and its *z* impact parameter is taken as the primary vertex position along *z*.

`STTrackSeeding`, using the sorted `SpacePoint` map and a Monte Carlo Look-Up Table (MC-LUT) linking each B-layer module to the ones belonging to other logical layers, builds track seeds formed by two `SpacePoints` and fits them with a straight line; one or more logical layers can be linked to the B-layer, the latter option being particularly useful if robustness to detector inefficiencies must be improved. If the primary vertex has already been reconstructed by `STMuonVertex`, a fraction of fake track seeds can be rejected during their formation, applying a cut on their *z* distance from the primary vertex. Otherwise, if no vertex information is available, an histogram whose resolution depends on the number of seeds found is filled with the *z* impact parameter of each seed; its maximum is then taken as *z* position for the

primary vertex. This vertexing algorithm, which can be operated in both RoI and full scan modes, is best suitable for high luminosity events containing many high p_T tracks (*e.g.*, b-tagging). Independent cuts on r-ϕ and z impact parameters are eventually applied to the reconstructed seeds to further reduce the fake fraction.

`STThreePointFit` extends track seeds with a third `SpacePoint`; it uses a Monte Carlo map associating to each seed a set of module roads [Footnote: A road is a list of modules ordered according to the radius at which they are placed starting from the innermost one] the track could have hit passing through the Pixel or SCT detectors. A subset of modules is extracted from each road according to a user defined parameter relating to their "depth" inside it (*e.g.*, the user can decide to use modules at the beginning or in the middle of each road, *etc.*). `SpacePoint`s from the selected modules are then used to extend the seed and candidate tracks are fitted with a circle; ambiguities (*e.g.*, tracks sharing at least one SpacePoint) can be solved on the basis of the track quality, leading to an independent set of tracks that can be used for trigger selection or as a seed for further extrapolation.

13.2.4.3 TRTLUT

TRT-LUT is a LVL2 tracking algorithm for track reconstruction in the TRT. It is described in detail elsewhere cite{ref:TRTLUT}. The algorithm takes as input Hits in the TRT. The algorithmic processing consists of Initial Track Finding, Local Maximum Finding, Track Splitting, and Track Fitting and Final Selection. It outputs the Hits used and Tracks with their parameters.

During the Initial Track Finding, every hit in a three-dimensional image of the TRT detector is allowed to belong to a number of possible predefined tracks characterized by different parameters. All such tracks are stores in a Look-Up Table (LUT). Every hit increases the probability that a track is a genuine candidate by one unit.

The next step consists of Local Maximum Finding. A two-dimensional histogram is filled with bins in ϕ and $1/p_T$. A histogram for a single track would consists of a “bow-tie” shaped region of bins with entries at a peak in the center of the region. The bin at the peak of the histogram will, in an ideal case, contain all the hits from the Track. The roads corresponding to other filled bins share straws with the peak bin, and thus contain sub-sets of the hits from the track. A histogram for a more complex event would consist of a superposition of entries from individual tracks. Hence, bins containing a complete set of points from each track can be identified as local maxima in the histogram.

The Track Splitting stage of the algorithm analyzes the pattern of hits associated to a track candidate. By rejecting fake candidates composed of hits from several low-p_T tracks, the track splitting step results in an overall reduction by a factor of roughly 2 in the number of track candidates. For roads containing a good track candidate, it identifies and rejects any additional hits from one or more other tracks. The result of the overall Track Splitting step is a candidate that consists of a sub-set of the straws within a road.

The final step of TRT-LUT, Track Fitting and Final Selection, performs a fit in the r-ϕ (z-ϕ) plane for the barrel (end-caps) using a third order polynomial to improve the measurement of ϕ and p_T. Only the straw position is used (*i.e.*, the drift time information is not used). The track is assumed to come from the nominal origin. After the fit, a reconstructed p_T threshold of 0.5GeV/c is applied.

13.2.4.4 TRTKalman

TRT-Kalman cite{ref:trtkal} is a new package based on xKalman++ (see Section [Ref: sect:xkal]). The name is in fact a misnomer since the Kalman filter component of xKalman++ is not used for the TRT; a histogram search and Least Squares fit is used instead.

TRT-Kalman incorporates following modified modules from xKalman:

- `XK_Tracker_TRT`: This reads TRT geometry from ROOT files. It uses `InDetDescr`, `InDetIdentifier` to access necessary Detector Description information;
- `XK_Algorithm`: A strategy is added to perform TRT standalone reconstruction;
- `XK_Track`: A step has been added with fine-tuning of track parameters after the histogramming step and Least Squares fit;
- `XKaTrtMan, XKaTRTRec`: This contains xKalman++ internal steering algorithms;
- `XKaTRTClusters`: This component retrieves `TRT_RDO_Container` from StoreGate filled from a ByteStream file.
-

[More text needed here from Sergey]

13.2.4.5 T2Calo

T2Calo cite{T2CaloCVS, T2CaloVariables, T2CaloRefSw, T2CaloRefSwStudies} is a clustering algorithm for electromagnetic (EM) showers, seeded by the LVL1 EM trigger RoI positions cite{RefLVL1}. This algorithm can select isolated EM objects from jets using the cluster *ET* and certain shower-shape quantities.

The RIOs are calibrated calorimeter cells (`LArCell`s and `TileCell`s), imported from the offline reconstruction. Both `LArCell`s and `TileCell`s have `CaloCell` as common base class. The output (`T2EMCluster`) is a specific LVL2 class containing the cluster energy and position, and the shower-shape variables useful for the selection of EM showers.

The first step in T2Calo is to refine the LVL1 position from the cell with highest energy in the second sampling of the EM calorimeter. This position ($\eta 1$, $\phi 1$) is later refined in the second sampling by calculating the energy weighted position (ηc, ϕc) in a window of 3×7 cells (in $\eta \times \phi$) centred in ($\eta 1$, $\phi 1$). In Ref. cite{T2CaloVariables}, the steps to perform the jet rejection are the following:

- In sampling 2, $Rshape\eta = E3 \times 7 / E7 \times 7$ is calculated. The expression $En \times m$ stands for the energy deposited in a window of $n \times m$ around ($\eta 1$, $\phi 1$). This shape variable takes into account that most of the energy of EM showers is deposited in the second sampling of the EM calorimeter.
- In sampling 1, $Rstrip\eta$ = ($E1$st - $E2$nd) / ($E1$st + $E2$nd) is obtained in a window of $\Delta\eta \times \Delta\phi = 0.125 \times 0.2$ around (ηc, ϕc). $E1$st and $E2$nd are the energies of the two highest local maxima found, obtained in a strip-by-strip basis. The two ϕ-bins are summed and only the scan in η is considered. A local maximum is defined as a single strip with energy greater than its two adjacent strips.
- The total transverse energy $E\mathrm{rm}\ T$ deposited in the EM calorimeter is calculated in a window of 3×7 cells around ($\eta 1$, $\phi 1$).
- Finally, the energy that leaks into the hadron calorimeter $EhadT$ is calculated in a window of size $\Delta\eta \times \Delta\phi = 0.2 \times 0.2$ around (ηc, ϕc).

13.2.4.6 muFast

The muFast algorithm is a LVL2 tracking algorithm for the Muon Spectrometer. In the past, it existed in the Reference software from ATRIG, and this version is described in detail elsewhere cite{ref:mufast}.

The algorithm itself consists of three parts:

- **Data Preparation**: This essentially rearranges simulated bits into what will come from the online system. MDT RDOs are put into data classes with electronics info and offline Identifiers. Use is made of a global map of MDT to retrieve global position of each DetectorElement.
- **Selection**: This consists of taking RDOs and converting them into a private structure using standalone code running with the RoI scheme. The processing steps are modular and substitutable. There is no data model for exchanging information, but instead, information is written into a common memory space. Use is made of a Look Up Table (LUT) which reads from an ASCII file.
- **Monitoring: This** component of muFast fills ntuples and histograms for use later on. It only needs the internal data structure filled in the Selection component of muFast.

13.2.5 HLT Algorithms for EF

13.2.5.1 xKalman++

xKalman++ is a package for global pattern recognition and Track fitting in the Inner Detector for charged tracks with transverse momentum above 0.5GeV/*c*. A more detailed description of this algorithm is available elsewhere cite{xKalman}.

The algorithm starts the track reconstruction in the TRT using a histogramming method or in the Pixel and SCT detector layers using segment search.

The first reconstruction method outputs a set of possible track candidate trajectories defined as an initial helix with a set of parameters and a covariance matrix. As a second step the helix is then used to define a track road through the precision layers, where all the measured clusters are collected. xKalman++ attempts to find all possible helix trajectories within the initial road and with a number of sufficient clusters.

The primary track finding in the Pixels or SCT outputs a set of SpacePoints as an initial trajectory estimation. In the next step these set of space points serve as an input for the Kalman filter-smoother formalism that will add the information from the precision layers. Each reconstructed track is then extrapolated back into the TRT, where a narrow road can be defined around the extrapolation result. All TRT Clusters together with the drift time hits found within this road are then included for the final track-finding and track-fitting steps.

There are three seeding mechanism available in the offline environment: `XKaSeedsAll`, the reconstruction of the full event; `XKaSeedKINE` reconstruction of a region of interest and soon available EM calorimeter seeding. In the HLT environment as an EF algorithm xKalman++ will be seeded by the LVL2 result.

After the pattern recognition and Track fitting steps xKalman++ stores the final Track candidates as `SimpleTrack` objects in a `SimpleTrackCollection`. The Track candidate contains the following information:

- Fit procedure used (m-fit or e-fit);
- Helix parameters and their covariance matrix at the end-points of the filter procedure in the precision layers (point on the trajectory closest to the vertex) and in the TRT (point on the trajectory closest to calorimeter);
- Total $\chi 2$ resulting from final fit procedure;
- List of all hits on track from all sub detectors;

- Total number of precision hits *Np*.
- Total number of straw hits *Ns*, empty straws crossed *Ne*, and of drift-time hits *Nt*.
-
- Furthermore, a track candidate is stored in the final output bank if it passes the following cuts:
- The number of precision hits is larger than 5 to 7;
- The ratio $Ns/(N\backslash s+Ne)$ is larger than 0.7 to 0.8;
- The ratio $Nt/N\backslash s$ is larger than 0.5 to 0.7;
- No previously accepted track has the same set of hits as the current one; this last cut removes full *ghost tracks*.

13.2.5.2 iPatRec

A detailed description of iPatRec is available elsewhere cite{iPatRec}.

[Need text here.]

13.2.5.3 LArClusterRec

LArClusterRec is the reconstruction package for electromagnetic clusters in the calorimeter cite{LArClusterRec}. It is generally used in conjunction with TileRec and referred to collectively as CaloRec.

In the first step towers are created by summing the cells of the electromagnetic calorimeter and the presampler in depth using a granularity of $\Delta\eta \times \Delta\phi = 0.025\times 0.025$. The input of the tower building are the calibrated calorimeter cells which are produced by the package `LArCellRec` or `TileRec`.

In the next step a sliding window algorithm is used. In case a local maximum is found with a total energy in the window above a given transverse energy threshold, clusters are created which are subsequently stored in the cluster container. To reconstruct the cluster energy and position is calculated in a given window. [Footnote: This window can be different from the one used for the sliding window algorithm.] The cluster energy is corrected for η and ϕ modulations and leakage outside the cluster in a given window. In the region between the barrel and end-cap calorimeter the cluster energy is in addition corrected for energy losses using the energy deposit in the crack scintillators. The η position in the first and second sampling is corrected for s-shapes, which is a geometrical effect. The ϕ position is corrected for an offset, which is also a geometry effect.

13.2.5.4 egammaRec

EgammaRec is designed to calculate useful quantities to separate clusters in the electromagnetic calorimeter from jets. To do so, electromagnetic cluster information as well as tracking information is used.

In the electromagnetic calorimeter electrons are narrow objects, while jets tend to have a broader profile. Hence, shower shapes can be used to reject jets. This is handled by the `EMShowerBuilder` which calls different algorithms which calculate diverse quantities using the information in the first and second sampling of the electromagnetic calorimeter as well as the leakage into the first sampling of the hadronic calorimeter.

Cluster and track information is combined in the `TrackMatchBuilder`. For a given cluster the nearest track is searched for by correctly taking into account the bending of tracks in the magnetic field. In case a track is found in a certain distance from the cluster and the E/p ratio fulfils $0.5 < E/p < 1.5$ the track match is successful. In the final e/jet separation step, jets can be rejecting by applying harder E/p cuts as well as a cut in $\Delta\eta$ and $\Delta\phi$ between the cluster and the track. The next final step after egammaRec has run is the final particle identification step. A first version of this package is available, but it is not yet tested.

13.2.5.5 Moore

Moore (Muon Object Oriented Reconstruction) is a track reconstruction package for the Muon Spectrometer. A detailed description of Moore is available elsewhere cite{Moore}.

Moore takes as input collections of digits or clusters inside the Muon Spectrometer (CSC, MDT, RPC, TGC) and outputs fitted reconstructed tracks whose parameters are expressed at the entrance of the muon spectrometer.

The reconstruction is performed in several steps and each step is driven by an Algorithm module, `MooMakeXXX`. Each algorithm is independent (*i.e.*, it retrieves objects created by the previous modules from StoreGate and it builds a transient object to be recorded in StoreGate where it is available for the subsequent algorithms). The only link between algorithms are the transient objects, in such a way that the algorithms depend on transient objects but transient objects do not depend on algorithms. The decoupling between data and algorithms and the natural step sequence of algorithm performing the reconstruction gives the opportunity to plug-in different reconstruction algorithms at run time.

As it is now, the overall reconstruction starts from the searches for ϕ regions of activity and builds `PhiSegments` (`MooMakePhiSegments`). For each ϕ-Segment, the associated MDTs are found and a *crude* `RZSegment` is built (this is essentially a collection of z hits) (`MooMakeRZSegments`).

Inside the MDTs the drift distance is calculated from the drift time, by applying various corrections: such as the TOF, the second coordinate, the propagation along the wire, the Lorenz effect. From the 4 tangential lines the best one is found. All the MDT segments of the outer station are combined with those of the Middle layer. The MDT hits of each combination are added to the phi-hits of the ϕ Segment, forming outer track candidates. All the successfully fitted candidates are kept for further processing (`MooMakeRoads`).

The successful outer track is subsequently used to associate inner station MDT hits. A final track is defined as a successfully fitted collection of trigger hits and MDT hits from at least two layers (`MooMakeTracks`). The parameters of the fitted track are referred to the first measured point and are therefore expressed at the entrance of the Muon Spectrometer.

When dealing with data already selected by the trigger the first two steps (`MooMakePhiSegments`) and (`MooMakeRZSegments`) can be substitute with *ad hoc* makers that seed the track search in the regions selected by the trigger.

13.3 Signatures, rates and efficiencies

In the following subsections, the physics performance of algorithms for LVL2 and EF is summarized for five final-state classes: electrons and photons; muons; jets, taus and missing ET; b-jets; and B-physics.

This broad classification stems from the physics goals of the ATLAS experiment, as explained in Chapter 4. Whenever possible, results will include the realistic use of data formats and associated converters (as described in previous section), steering control (as described in Chapter 9), highlighting the flexible boundary between LVL2 and EF. Selection schemes are then derived, which contain the signatures used to decide whether or not to reject events. In order to maximize the discovery potential, the selection schemes generally only use inclusive signatures. Except for the case of B physics, reconstruction of exclusive decays is not required and no topological variables (e.g. the calculation of invariant masses from a combination of several high-p_T objects) are used in the selection, although this is technically feasible at LVL2 or in particular in the EF (e.g. to select $Z \rightarrow l^+l^-$ decays exclusively).

It is worthwhile noticing that system performance (*e.g.* execution time, amount of data needed) is one of the major requirement in the HLT selection, to comply with the constraints imposed by the on-line environment and resources. In this chapter an indication of the compliance with those requirements will be given for the most important selections, whilst a detailed analysis of the different contributions to those figures will be given in Chapter 15. In general, all results have been achieved by optimizing concurrently physics and system performances.

13.3.1 e/gamma

In the present view of the ATLAS trigger menus, the inclusive electron and photon triggers are expected to contribute an important fraction of the total high-p_T trigger rate. After the selection in LVL2 and the EF, the remaining rate will contain a significant contribution from signal events from Standard Model physics processes containing real isolated electrons or photons ($W \rightarrow ev$, $Z \rightarrow ee$, direct photon production, etc.).

The electron and photon triggers can be viewed as a series of selection steps of increasing complexity. After receiving the LVL1 electromagnetic (e.m.) trigger RoI positions, the LVL2 trigger performs a selection of isolated e.m. clusters using the full calorimeter granularity and detailed calibration. This selection is based on cluster E_T and shower-shape quantities that distinguish isolated e.m. objects from jets. A further, more refined calorimeter-based selection may classify the e.m. cluster as a LVL2 photon trigger object.

Electrons are identified at LVL2 by associating the e.m. cluster with a track in the Inner Detector. This association can be as simple as requiring the presence of a track with a minimum p_T in the e.m. RoI, but may, in addition, require position and momentum matching between the track and the cluster. Typically, track candidates are found by independent searches in the TRT (*needs to be seen if we get it working in time*) and SCT/Pixel ('Precision') detectors in the region identified by the LVL1 RoI. Details of the different LVL2 algorithms are described in.

As currently planned by the HLT scheme, the EF will select events using as far as possible the algorithms of the ATLAS offline reconstruction system. The present study therefore uses the available ATLAS offline reconstruction software as a prototype of the future EF code. However, the criteria to identify electrons and photons are softer at the EF level in order not to loose events prematurely. The EF algorithm components (calorimetry, tracking and particle identification) are treated in a similar way as for LVL2. The main differences with respect to LVL2 derive from the availability at the EF of more detailed calibrations and more sophisticated algorithms with access to the full-event data. The improved performance results in sharper thresholds and better background rejection. In the case of electrons, bremsstrahlung recovery will be performed for the first time at the EF. In addition, a photon-conversion recovery procedure will be applied to photon candidates at the EF.

In the following the system and physics performance of the selection of electrons by the HLT will be reviewed in detail. The photon selection will not be considered here and these study are currently in progress. The physics performance of the electron and photon selection has been already studied in detail in the past by the HLT and are reported in the trigger TP. The system performance of this selection is discussed in ***Chapter 15.***

13.3.1.1 HLT Electron Selection Performance

The performance of the electron and photon triggers has been estimated for single electrons and photons, and for some standard physics channels (e.g. $Z \rightarrow ee$, $W \rightarrow e\nu$, $H \rightarrow 4e$). The performance has been characterized in terms of efficiency for the signal channel, rate expected for the selection and algorithm execution time. The rates shown in this and in the following sections have been obtained using a sample of simulated di-jet events with pile-up added for the low and design luminosity scenario. In general, events with electrons and photons are selected on the basis of single high-p_T objects or of pairs of lower-p_T objects. The physics performance of the electron triggers is summarized here and documented in detail for both the LVL2 trigger, and the EF, algorithms in XXX.

Table 13-1 Performance of the isolated electron HLT trigger at design and low luminosity for the single electron. The results are presented in a single sequence, except for the starting point of the LVL2 tracking, where two alternatives (TRT and Precision) are shown. 'Matching' refers to position and energy–momentum matching between calorimeter clusters and reconstructed tracks (at LVL2 both Precision and TRT tracks are used). The efficiencies are given for single electrons of p_T = 30 (25) GeV a design (low) luminosity over the full rapidity range $|\eta| < 2.5$. The efficiencies and rates are given with respect to a LVL1 output efficiency of 9x% (9x%) and a LVL1 rate for e.m. clusters of xxx kHz (xxx kHz). The timing results quoted here are for events from the di-jet sample and are scaled to correspond to a 500 MHz Pentium III machine running Linux (what will be our reference? Answer: It will be a 4 GHz machine, Valerio). The terms m_{50} and m_{95} are defined in xxx. The quoted errors are statistical.

	Design Luminosity			Low Luminosity		
Trigger Step	**Rate [Hz]**	**Efficiency [%]**	**Timing m_{50} / m_{95}**	**Rate [Hz]**	**Efficiency [%]**	**Timing m_{50} / m_{95}**
LVL2 Calo	3490 ±± 160	97.1 ± 0.3	0.20 / 0.26 ms	1100 ± 30	96.0 ± 0.6	0.15 / 0.23 ms
LVL2 Precision	620 ± 70	90.3 ± 0.6	6.2 / 12.7 ms	150 ± 11	92.4 ± 0.8	2.4 / 5.8 ms
LVL2 TRT	1360 ± 100	89.7 ± 0.6	0.4 / 1.2 s	360 ± 17	89.2 ± 0.9	31 / 210 ms
LVL2 Matching	460 ± 60	85.3 ± 0.7	--	140 ± 11	88.1 ± 0.9	--
EF Calo	313 ± 50	83.5 ± 0.8	0.39 / 0.63 s	85 ± 8	86.4 ± 1.0	0.34 / 0.56 s
EF ID	149 ± 34	79.3 ± 0.8	11 / 71s	57 ± 7	82.4 ± 1.1	0.31 / 1.6 s
EF Matching	117 ± 30	77.6 ± 0.8	--	41 ± 6	80.8 ± 1.2	--

The performance of the single isolated electron HLT algorithm is summarized in table xxx as a function of the main steps in the LVL2–EF trigger chain. The trigger steps have been factorized by detector in order to show the overall computational load and rejection that each stage contributes to the trigger. The table shows that the input rate from the LVL2 electron trigger to the EF is xxx Hz (xxx Hz) at design (low) luminosity for a nominal p_T threshold of 30 GeV (25 GeV). The overall reduction in rate achieved by LVL2 is a factor of xx (xx) for a loss of efficiency of xx% (xx%) with respect to LVL1. The additional

rate reduction provided by the EF amounts to a factor of xxx (xxx) for a relative efficiency loss of xxx% (xxx%). Using the offline electron selection the rate is reduced by xxx (xxx)%, for an additional loss of xx (xx)% in efficiency. This shows that the HLT selection is very powerful. The LVL2 selection has an efficiency of 9x% (9x%) for the events selected by the EF alone, and the additional loss of events is mostly due to the fast track selection at LVL2, showing the expected correlation of inefficiencies at the LVL2 and EF stages (e.g. due to bremsstrahlung).

At low luminosity, the events remaining after the HLT electron selection consist of $W \rightarrow e\nu$ decays (xx ± x)%, isolated electrons from $(b,c) \rightarrow eX$ decays (xx ± x)% and background from high-p_T photon conversions and misidentified hadrons (xx ± x)%. At design luminosity, where a higher p_T threshold is applied, the corresponding proportions are (xx ± x)%, (xx ± x)% and (xx ± x)%. The quoted errors are the statistical uncertainties on the estimates. As seen around 50% of the selected events contain 'real' electrons, hence a further improvement of this selection can only be small.

Electron decays of the W are selected by the EF with an efficiency of (xx ± x)% at low luminosity and (xx ± x)% at design luminosity, in agreement with the values given in table xxx for single electrons of 25 GeV and 30 GeV transverse momentum respectively. Finally, as an example of the performance for a physics signal, the HLT selection efficiency (using both the single- and the double-electron trigger) for the decay $H(130) \rightarrow 4e$ is (9xx ± x)% with respect to the LVL1 efficiency of 9x.x% at low luminosity; these high efficiencies are due to the large electron multiplicity in the final state.

13.3.1.2 HLT Electron/Photon Algorithm Optimization

The algorithm execution time has been measured in order to study the resource constraints they may place on the overall HLT/DAQ system. This exploratory study addresses the interplay between the physics and the system performance aspects. Timing measurements were carried out on the feature-extraction part of the algorithms, excluding as much as possible any I/O (data read/write), and thus characterizing the most computationally complex aspects of the algorithms. In order to assess the impact of tails on the timing results, the measurements are given in terms of the median (m_{50}) and the latency within which 95% of the events are processed (m_{95})[1].

To understand where the computing resources are being used in the trigger, the different parts of the LVL2 and EF algorithms have been profiled in the test-bed studies, which are summarized in SAUL's section. There are still quite some possibilities to speed up the execution time and the numbers given in section xxx will improve with time. Another possibility to speed up the code is to optimize different algorithm parameters. Here the aim is to eliminate any resource-consuming tasks that contribute only marginally to the rejection. To give an example, in case zero-suppression is applied for the calorimeter cells, the execution time for data access as well as execution of the LVL2 electron selection by T2Calo can be reduced. Applying a zero-suppression cut of 3σ of the electronic noise level reducing the time for the algorithm from xxx (xxx) to xxx (xxx) ms and for the data access from xxx (xxx) to xxx (xxx) at low and design luminosity.

Similar studies have been performed for the EF. As an example, Figure xxx shows the execution-time dependence of the EF electron-tracking algorithm on the transverse-energy threshold of the calorimeter cluster used to *seed* the reconstruction. (The seed energy scale does not correspond to the calibrated electron energy scale.) Increasing the threshold, thus reducing the number of seeds, reduces the execution time (in particular m_{95}) with a negligible impact on the physics performance.

1. The timing measurements were carried out on several different platforms, but have been converted to the same overall scale, corresponding to a 500 MHz Pentium II equivalent.

The present system performance of the electron/photon algorithms can be improved at all levels of the HLT. There are studies under way which will be documented in future TDAQ notes.

(Comment from Monika: In this section I just want to give few examples and some outlook of what can be expected in the future. Boundary with Saul's section has to be sorted out.)

13.3.1.3 HLT Strategy and the LVL2–EF Boundary

The use of system resources in the electron HLT can be minimized by exploiting the modularity of the trigger. By ordering the trigger steps in such a way that events are rejected as early as possible, both overall processing times and data transfers are reduced. As an example, at design luminosity, the EF total electron trigger execution time is reduced by a factor of two by rejecting events immediately after the calorimeter reconstruction. Similar gains are possible at LVL2.

Factorizing the trigger algorithm components also provides flexibility to move the rejection power from LVL2 to the EF or vice versa, to optimize the following: the performance of the implementation of the algorithm; the robustness of the selection with respect to the rate; the load implied at each level; etc. As an example, figure xxx shows that an increase in efficiency can be obtained, with a modest increase in the total HLT output rate, by moving the whole LVL2 tracking selection to the EF. However, in this case, the input rate to the EF would increase by a factor of about eight, with important consequences on the computing load on the EF.

An important aspect of optimizing the sharing of rejection between LVL2 and the EF is the determination of the rejection contributed by each trigger level at the same efficiency. After tuning the LVL2 and EF electron selections to yield the same efficiency for events selected by LVL1, the EF contribution to the total reduction in rate is still better than LVL2 by a factor of two (three) at design (low) luminosity

as explained in STEFAN's section.

In case the incoming trigger rate is too high and needs to be reduced two obvious ways to do so is either to raise the energy threshold of the trigger menu item or by stricter selection criteria. All of these will imply an additional loss in efficiency for physics signals. Part of this loss in physics can then be recovered by more selective triggers. The preferred and easiest way to reduce the rate is to raise the energy thresholds. The LVL1 rate is dominated by the contribution from single high-p_T e.m. objects. As an example, raising the thresholds by E_T=5GeV of the single electron trigger would yield in a final HLT rate of xxx (xxx) at low (design) luminosity. This is also seen in Figure xxx, which illustrates the impact of raising the threshold for the single-electron HLT selection only (nominal threshold of 30 GeV), while keeping the double-electron trigger threshold at its nominal value (20 GeV for each electron). The upper plot indicates the reduction in rate for the sum of the single- and the double-electron trigger contributions. As the threshold is increased, besides the reduction of fake electrons, also the contribution from real $W \rightarrow e\nu$ decays is gradually rejected. The lower plot shows the impact on another physics signal, the $Z \rightarrow e^+e^-$ decay: for thresholds below 35 GeV, the efficiency for Z's is only slightly reduced. Decays with more than two electrons are affected even less, e.g. in the case of $H(130) \rightarrow 4e$.

As illustrated above, the proposed strategy contains considerable flexibility. Various possibilities exist to reduce the required computing resources or to improve the physics performance. For many channels of interest, the selection scheme also provides considerable redundancy.

13.3.2 Muon selection

The main purpose of the high-level muon trigger is the accurate reconstruction of muon tracks in the RoIs indicated by the LVL1 muon trigger. LVL2 and EF must reject low-p_T muons, secondary muons produced in the in flight decays of charged pions and kaons and fake muons originating from the cavern background. The EF must be able to reconstruct additional muons present in the event not reconstructed or selected by the LVL2 trigger.

Whilst the LVL1 trigger system uses only hits from the dedicated trigger detectors (RPCs in the barrel and TGCs in the endcap), the LVL2 and EF has access to the full measurements of the Muon Spectrometer, including in particular the data from the Monitored Drift Tubes (MDTs). This allows the best muon track reconstruction. The high background environment in the Muon Spectrometer demands algorithms with robust and fast pattern recognition capable of rejecting hist induced by the cavern background.

The tracks found in the LVL2 Muon Trigger are extrapolated for combination with the Inner Detector and the Calorimeter. Matching between muon tracks measured independently in the Muon System and the Inner Detector selects prompt muons and reject fake and secondary muons. This is important in particular for the B-physics trigger in low-luminosity running, for which the selection of prompt low-p_T muons events defines the input of the B-physics trigger algorithm.

The studies presented in this section are limited to the barrel region ($|\eta|<1$).

13.3.2.1 The LVL2 Muon Standalone Algorithm muFast

The muFast algorithm has been designed expressly for the online selection environment. The program is steered by the RoI given by the LVL1 Muon Trigger and uses both RPCs and MDTs measurements. At present this algorithm is limited to the barrel region and it is based on four sequential steps:

1. LVL1 emulation; the muon pattern recognition in the MDT system is initiated by the RPC hits that induced the LVL1 trigger accept. Among these hits, only those related to the pivot plane (middle RPC station) are provided by the muon trigger processor; the ones related to the coincidence plane (innermost and outermost RPC stations) have to be identified running a fast algorithm that simulates the basic logic of the LVL1selection.
2. Pattern recognition: it is performed using the RPC hits that induced the LVL1 trigger to define a road in the MDT chambers around the muon trajectory. MDT tubes lying within the road are selected and a contiguity algorithm is applied to remove background hits not associated with the muon trajectory;
3. A straight-line track fit is made to the selected tubes (one per each tube monolayer) within each MDT station. For this procedure the drift-time measurements is used to fully exploit the high measurement accuracy of the muon tracking system. The track sagitta is then evaluated.
4. A fast p_T estimate is made using LUTs. The LUT encodes the linear relationship between the measured sagitta and the Q/p_T, as a function of eta and phi.

The output of this algorithm is the measurement of the muon transverse momentum p_T at the main vertex, eta and phi.

13.3.2.2 The LVL2 Muon Combined Algorithm muComb

The combination of the features of the track measured in the Muon Spectrometer and the Inner Detector (ID) at LVL2 provides a rejection of π and K decays to μ and of fake muons induced by the cavern back-

ground. Moreover the combination of the two measurements improves the momentum resolution of reconstructed muons over a large momentum range.

The matching of the Muon Spectrometer tracks and of the ID can be performed extrapolating the ID track to the muon system. The procedure needs to take into account the detector geometry, the material composition and the inhomogeneity of the magnetic field. An accurate extrapolation would require the use of detailed geometry and magnetic field databases, together with a fine tracking. All this would is expensive in terms of CPU time and therefore not acceptable for the LVL2 trigger.

To provide a fast tracking procedure, the effects of the geometry, the materials and of the magnetic field have been described by simple analytic functions of eta and phi. The extrapolation of the ID tracks to the entrance of the Muon Spectrometer is performed using linear extrapolation in two independent projections: the transverse and the longitudinal views. Two coordinates are extrapolated: the z-coordinate and the azimuthal angle phi. The linear extrapolation is corrected using average corrections. In the transverse projection the ID track extrapolation in phi is corrected as follows:

$$\Delta\varphi = \frac{\alpha}{p_T - p_T^0} \qquad 13\text{-}1$$

where α is related to the field integral and p_T^0 allows for the transverse energy loss in the material of the calorimeter, that is approximately independent of the track transverse momentum p_T. Both alpha and p_T^0 have been determined by fitting $\Delta\varphi$ of simulated muons as a function of p_T. It is found that $p_T^0 \sim 1.5$, *i.e.* about half of the transverse energy loss of low energy muons, as naively expected. A similar approach has been followed in the case of the extrapolation of the z-coordinate in the longitudinal view.

The matching is done geometrically using cuts on the residuals in each of z and phi.

For matching tracks the combined transverse muon momentum is estimated through a weighted average of the independent p_T measurements in the Muon Spectrometer and in the Inner Detector. For each combined track, a χ^2 parameter is used to evaluate the quality of the p_T matching. Thanks to the high quality of the muon p_T measurements in both detectors, secondary muons from π and K decays give typically bad χ^2 matching, and thus can be rejected.

13.3.2.3 The Muon Event Filter Algorithm MOORE

in preparation

13.3.2.4 The Physics Performances of LVL2 Muon algorithms

The p_T resolution of reconstructed muons is crucial to the selection efficiency and to the rejection of low p_T tracks that can be achieved at LVL2. The distribution of $(1/p_T^{\mathrm{muon}} - 1/p_T^{\mathrm{true}})/(1/p_T^{\mathrm{true}})$ obtained by the muFast algorithm is shown in figure [figure TP 8-5] for p_T=6 GeV/c. The non Gaussian tails arise largely from the presence of soft particles produced by the muon interacting with the material of the detector.

The p_T resolution of the muFast algorithm is shown as a function of p_T in figure [figure TP 8-7]. As shown in the figure, the resolution ranges between 4.0% and 5.5% for muon in the p_T interval 6-20 GeV/c. These results are well compared with the transverse momentum resolution obtained by the offline muon reconstruction program MUONBOX.

The selection efficiency of muFast for selecting prompt single muons at 6 GeV/c and 20 GeV/c thresholds, relative to muons accepted by the LVL1 muon trigger, are shown in figure [figure TP 8-9]. For a nominal threshold of 6 GeV/c, the efficiency is about 90%, including detector acceptance. This efficiency is 95% for the 20 GeV threshold.

The total rates after this algorithm, including the rejection provided by the LVL1 selection, have been evaluated by convolving the algorithm efficiency as a function of p_T with the muon differential cross section production of the dominant physics processes. Where the available statistics are too low (in particular for the high-p_T rate calculation) to evaluate the efficiency, the lowest p_T at which an efficiency estimate has ben possible (p_T=10 GeV/c) is assumed conservatively to constitute a plateau extending down to the lower limit of the p_T acceptance (p_T=3 GeV/c in the barrel). The rates from π/K decays are calculated using the predicted cross-sections from the DMPJET program, and would be lower by about 50% if the PYTHIA prediction were used.

Table 13-2 Total output rates [kHz] of the LVL2 muon trigger after application of the mu rithm for the 6 GeV/c low-p_T threshold at low and 20 GeV threshold at the design luminosity.

Physics Process	low-p_T	high-p_T
p/K decays	3.1	0.06
b decays	1.0	0.09
c decays	0.5	0.05
W→μν	negligible	0.05
cavern background	negligible	negligibl
Total	**4.6**	**0.24**

The total rates after LVL2 are shown in Table 13-2. [THE ABOVE NUMBERS NEEDS TO BE CHECKED].

Preliminary studies of the trigger rate arising from the cavern background as predicted by the FLUKA package have been done. The probability that a fake LVL1 muon trigger is accepted by the LVL2 is below 10^{-2}. This upper limit is sufficient to neglect the contribution from fake muons.

Preliminary studies have been made to evaluate the physics performances of the muComb algorithm. Figure [figure TP 8-10] shows the combined reconstruction efficiency of prompt and secondary muons, as a function of the muon p_T, where the standalone codes from muFast and the LVL2 Precision algorithm [reference to the related chapter] have been used. The requirement of a good muon track matching (z/phi and p_T matching) reduces the low p_T trigger rate to 1.0 kHz: a factor three reduction compared to the rate from the muFast algorithm. Including the further reduction in rate due to the increase in p_T resolution for prompt muons, the total rate from the muComb algorithm is 2.1 kHz from muons with p_T>6 GeV/c.

13.3.2.5 The Physics Performances of the Muon Event Filter

in preparation

13.3.2.6 The Timing Performances of the Muon Algorithms

The muFast trigger algorithm has been benchmarked on several processor. On a processor corresponding to 10 SPECint95, muFast takes ~2ms/RoI, fairly independent from the trigger threshold and the muon p_T analyzed. If the data access is taken into account the time increases to XX ms.

A realistic evaluation of the time needed to the LVL2 muon trigger to take a decision has to take into account the time needed to move the data from the RoBs to the LVL2 processor in the ATLAS TDAQ architecture. Testbed measurements have been performed and indicates that the global time taken from the LVL2 trigger is about YY ms.

13.3.3 Tau/jets/$E_T$miss

Highlight major discovery channels, for taus probably start using bricolage

13.3.4 b-tagging

Define on-line strategy for this, explain why we think we need it, discuss implications for jets thresholds (and hence rates)

13.3.5 B-physics

About one collision in every hundred will produce a *bb* quark pair. Therefore, in addition to rejecting non-*bb* events, the B-trigger must have the ability to identify and select those events containing B-decay channels of specific interest. Important areas include CP-violation studies with the channels $B_d \rightarrow \pi^+\pi^-$ and $B_d \rightarrow J/\psi K_s$ (with both $J/\psi \rightarrow e^+e^-$ and $J/\psi \rightarrow \mu^+\mu^-$); measurements of B_s oscillations in $B_s \rightarrow D_s\pi$ and $B_s \rightarrow D_s a_1$ with $D_s \rightarrow \pi\pi$; analysis of $B_s \rightarrow J/\psi$ and $B \rightarrow J/\psi\eta$; rare decays of the type $B_{d,s} \rightarrow \mu^+\mu^-X$; *b*-production measurements and precision measurements with *B*-hadrons. Since these are precision measurements and searches for rare decays, high statistics are required. The large number of *bb* pairs produced at the LHC mean that ATLAS is well placed to make a significant contribution in these areas.

Since the Technical Proposal the B-trigger has been re-assessed in the light of a number of developments, including the likelihood of a reduced ID layout at the start of running, an increase in the target start-up luminosity and various trigger deferral scenarios. The aim is to provide the maximum possible coverage of key B-physics channels within the available resources.

It is important to study a range of scenarios since the actual start-up conditions are uncertain, luminosity is expected to vary from fill-to-fill, and there are uncertainties in the physics cross-sections and in the calculation of required resources. A flexible trigger strategy has, therefore, been developed based on a di-muon trigger at the start of higher luminosity LHC fills and introducing further triggers later in the beam coast or for lower luminosity fills (over the period of a beam-coast the luminosity is expected to fall by about a factor of two). Two strategies have been investigated for these additional triggers, as follows.

- Require a LVL1 JET or EM RoI in addition to a single-muon trigger (p_T>8 GeV). At LVL2 and the EF, tracks are reconstructed within RoI using pixel, SCT and TRT information. The reconstructed tracks form the basis of selections for e.g. $J/\psi(ee)$, $B(\pi\pi)$ and $D_s(\phi\pi)$. Since track reconstruction is performed inside RoI, the resources required are modest.
- A full-scan of the SCT and pixels is performed for events with a single-muon trigger (p_T>8~GeV). The reconstructed tracks form the basis of selections for e.g. $B(\pi\pi)$ and $D_s(\phi\pi)$. This promises better efficiency than the above method, but requires somewhat greater resources in order to perform the full-scan.

In all cases, at least one level-1 muon trigger is required to initiate the B-trigger. Since the cross-section for inclusive muon production from pion and kaon decays falls more rapidly with p_T than that for prompt muon production from *b*-decays, an appropriate choice of p_T threshold gives a powerful reduction of the trigger rate due to background processes. For example, a threshold of p_T > 8 GeV would gives a single-muon trigger rate of 10 kHz at LVL1 for a luminosity of 10^{33} $cm^{-2}s^{-}1$. Most of this rate is due to muons with true p_T below threshold originating from pion and kaon decay, a large proportion of which can be rejected at LVL2 on the basis of more precise track measurements. After the LVL2 selection the trigger rate is about 2~kHz at a luminosity of 10^{33} $cm^{-2}s^{-}1$; about one third of this rate is due to b→μ decays. It is important not to set the muon p_T threshold too high as this would significantly reduce the statistics in the signal channels and render the measurements un-competitive.

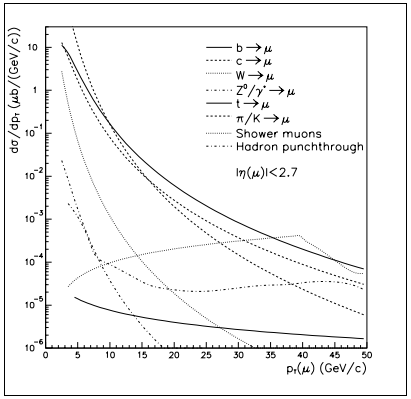


Figure 13-1 Differential cross-section ds/dp_T for inclusive muon production in ATLAS in the pseudo-rapidity range |η| < 2.7.

13.3.6 Di-muon triggers

A di-muon trigger provides a very effective selection for several important channels, e.g. $B \rightarrow J/\psi(\mu^+\mu^-)K_s$ and $B \rightarrow \mu^+\mu^-(X)$. The LVL1 muon trigger is efficient down to a p_T of about 5 GeV in the barrel region and about 3 GeV in the end-caps. However the actual thresholds used for the di-muon trigger will be determined by rate limitations. For example, a p_T threshold of 6 GeV would give a di-muon trigger rate of about 600 Hz after LVL1 at a luminosity of 2.0×10^{33} $cm^{-2}s^{-1}$. These triggers are mostly due to muons from heavy flavour decays plus some single muons which are doubly counted due to overlaps in the end-cap trigger chambers. The later are removed when the muons are subsequently confirmed at LVL2 using information from the muon precision chambers and ID. At the EF, tracks are refit and specific selections made on the basis of mass and decay length cuts. These consist of semi-inclusive selections, for example to select $J/\psi(\mu^+\mu^-)$ decays with a displaced vertex, and in some cases exclusive selections such as for $B_{d,s} \rightarrow \mu^+\mu$. The final trigger rate, after the EF, is about 20~Hz at a luminosity of 2.0×10^{33} $cm^{-2}s^{-1}$.

13.3.7 Hadronic final states

For hadronic final states, two strategies have been studied based on, for events with a muon trigger, either an ID full-scan or a RoI-based selection. An ID full-scan consists of track-reconstruction within the entire volume of the SCT and Pixel detectors <Ref_idscan> and, optionally, the TRT <ref TRTscan>. The alternative strategy uses low E_T LVL1 jet clusters to define RoIs for track reconstruction in the ID. By limiting track reconstruction to the part of the ID lying within the RoI, about 10% on average, there is potential for up to a factor of ten saving in execution time compared to the full-scan. Preliminary studies of efficiency and jet-cluster multiplicity have been made using a fast simulation which includes a detailed parametrization of the calorimeter. These studies indicate that a threshold of E_T > 5 GeV gives a reasonable jet cluster multiplicity, i.e. a mean of about two RoI per event for events containing a muon with p_T > 6 GeV, see Fig. <jet roi Mult>. A trigger based on this threshold would be efficient for $B \rightarrow D_s\pi$ and $B \rightarrow \pi\pi$ events with a *B*-hadron p_T above about 15 GeV.

Following the ID track reconstruction (either full-scan or RoI-based) further selections are made for specific channels of interest. These are kept as inclusive as possible at level-2 with some more exclusive selections at the EF. For example, samples of $B_s \rightarrow D_s\pi^+$ and $B_s \rightarrow D_s a_1$ events can both be triggered by selecting events containing a $D_s(\phi\pi^-)$ candidate.

Tracks are refit at the EF inside RoI defined from the results of LVL2. Using LVL2 to guide the EF reconstruction reduces the amount of data to be processed. For example, a region encompassing all LVL2 tracks forming $D_s(\phi\pi)$ or $B(\pi\pi)$ candidates corresponds to about 10% of the ID acceptance, on average. At the EF, tighter mass cuts may be applied than at LVL2, due to the better track parameter resolution obtained from the EF reconstruction. In addition, EF selections may include decay vertex reconstruction, allowing further cuts on vertex-fit quality and decay length.

Studies using a full detector simulation have shown that an efficiency of about 70% can be obtained for $B_s \rightarrow D_s\pi$ signal events where all final state particles have $p_T > 1.5$ GeV. The corresponding trigger rate is about 60 Hz at LVL2 and about 6 Hz after the EF at a luminosity of 10^{33} cm^{-2}s^{-1}, using a single muon trigger threshold of $p_T > 8$~GeV. There is very little degradation of the trigger performance if the number of pixel layers is reduced from three to the two layers expected at the start of LHC running.

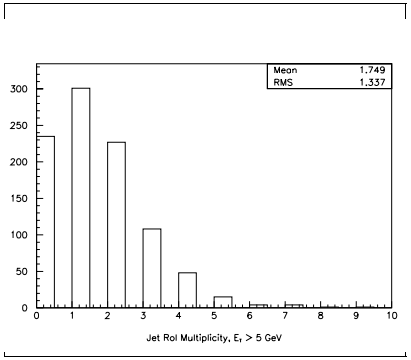


Figure 13-2 Jet RoI multiplicity ($E_T > 5$ GeV) for events with a muon $p_T > 6$ GeV.

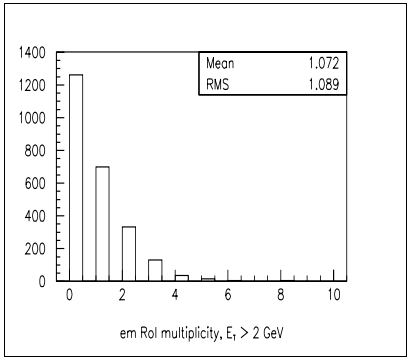


Figure 13-3 EM RoI multiplicity ($E_T > 2$ GeV) for events with a muon $p_T > 6$ GeV.

13.3.8 Muon-electron final states

A muon-electron trigger is used to select channels such as $B_d \rightarrow J/\psi(e^+e^-)K_s$ events with an opposite side muon tag, or $B_d \rightarrow J/\psi(\mu^+\mu^-)K_s$ with an opposite side electron tag. As for the trigger for hadronic final states, two different strategies have been studied using either an ID full-scan or RoI-based ID track reconstruction. In both cases a level-1 muon trigger, confirmed at level-2, is required.

For the full-scan based method, a histogramming technique <Ref_TRTLUT,Ref_xKalman> is used to search for tracks within the entire volume of the TRT. Good efficiency has been obtained for electrons with p_T down to about 1 GeV. However, since execution time scales as $1/p_T$, in practice higher thresholds may be used. To improve track parameter resolution, track candidates reconstructed by the TRT are then extrapolated into the SCT and pixels using a Kalman filter algorithm <Ref_SiKalman>. The TRT identifies e^+/e^- candidates on the basis of transition radiation information. Candidates passing track cuts are combined in opposite charge-sign pairs and $J/\psi(ee)$ mass cuts applied. An efficiency of about 40% can be obtained for $B_d \rightarrow J/\psi(e^+e^-)K_s$ events where both e^+ and e^- have $p_T > 1$~GeV. The corresponding LVL2 trigger rate is about 40 Hz, at a luminosity of 10^{33} cm^{-2}s^{-1}, using a $p_T > 8$~GeV muon trigger threshold. The tracks are refit at the EF, including a vertex fit. Decay length and fit quality cuts are applied, giving about a factor of ten further reduction in trigger rate.

An alternative strategy is based upon using the LVL1 trigger to find low E_T electron/photon clusters which define RoI to be investigated at LVL2. Preliminary studies, using a fast simulation, show that a reasonable compromise between RoI multiplicity and electron efficiency might be obtained with a threshold of E_T>2 GeV. This gives a mean RoI multiplicity of about one for events containing a muon with $p_T > 6$~GeV, see Fig. <EM RoI mult>. The corresponding efficiency to create a RoI for both the e^+ and e^- from $J/\psi(e^+e^-)$ is about 80% in events where both final state particles have $p_T > 3$~GeV. At LVL2, the electron/photon RoIs are confirmed in the calorimeter, using full-granularity information and including the pre-sampler. A search is then made, inside the RoI, for tracks in the SCT, Pixels and TRT. The RoI about each electron candidate can be quite small, of order $\Delta\eta \times \Delta\phi = 0.2\text{x}0.2$. This gives a large saving in reconstruction time, compared to a full-scan, but has a lower efficiency, particularly at low p_T.

13.3.9 Resource estimates

In order to estimate the computing resources required for the *B*-trigger, measurements of execution time are combined with estimates of trigger rate at each step of the selection. Various reconstruction algorithms have been timed on several different platforms in order to determine the mean execution time at a given luminosity, and the scaling of execution time with the number of hits in an event, and hence the scaling with luminosity. These timing measurements have been combined with the estimates of trigger rates and RoI multiplicity to give an estimate of the resources required for the B-trigger Ref. <Ref. B-trigger resources>. The results are shown in Table <resource table>.

Table 13-3

Luminosity	B-Trigger	no. cpu
$2\text{x}10^{33}$ cm^{-2}s^{-1}	Di-muon only	2
10^{33} cm^{-2}s^{-1}	Di-muon + RoI-based triggers	8
	Di-muon + fill-scan based triggers	26

The use of low E_T level-1 RoI to guide reconstruction at level-2 promises a full programme of B-physics for very modest resources. However multiplicities and efficiencies need to be verified in studies using a full detector simulation.

13.4 Event rates and size to off-line

Define present ideas about data compression and reduction, zero suppression for LAr (and TRT?): this might be probably be elsewhere as well. Differences between zeros at the EF and loss-less data compression in the ROSes.

Global table on rates for initial and high luminosity, implication for off-line reconstruction (costing, later)

13.5 Start-up scenario

Should be here? Picture a global approach on how we are going to handle, at the selection level, the first year of running, assuming a certain machine scenario. It is probably very appealing for LHCC

13.6 References

13-1 *ATLAS detector and physics performance technical design report*, CERN–LHCC/99–14/15 (1999)

13-2 S. Armstrong et al., “Requirements for an Inner Detector Event Data Model”, ATLAS-TDAQ-2002-011.

13-3 [ref:rpc2]K. Assamagan et al., “A Hierarchical Software Identifier Scheme,” ATLAS-COM-MUON-2002-019.

13-4 [ref:IoVS]C. Leggett and A. Schaffer, presentations at the ATLAS EDM-DD Workshop, 27 January 2003.

13-5 [SCTKalman]For more information on SCTKalman see P. Billoir and S. Qian, Nucl. Instr. Meths., A294 (1990) 219; P. Billoir and S. Qian, Nucl. Instr. Meths., A295 (1990) 492; I. Gaines, T. Huehn, and S. Qian, in Proceedings of CHEP97 (Berlin); I. Gaines and S. Qian, in Proceedings of CHEP98 (Chicago); I. Gaines, S. Gonzalez and S. Qian, in Proceedings of CHEP2000 (Padova); D. Candlin, R. Candlin and S. Qian, in Proceedings of CHEP01 (Beijing); J. Baines, et al. ATL-DAQ-2000-031.

13-6 [ref:TRTLUT]J. Baines et al., “Global Pattern Recognition in the TRT for B-Physics in the ATLAS Trigger”, ATLAS-TDAQ-99-012. M. Sessler and M. Smizanska, “Global Pattern Recognition in the TRT for the ATLAS LVL2 Trigger”, ATLAS-TDAQ-98-120.

13-7 [ref:trtkal]S. Sivoklokov, presentations made in PESA Core Algorithms Group meetings, December 2002 and January 2003. See also S. Sivoklokov, “High pT Level 2 Trigger Algortihm for the TRT Detector in ATRIG”, ATLAS-TDAQ-2000-043.

13-8 [T2CaloCVS]M.P. Casado, S. González, and T. Shears, TrigT2Calo package, http://atlas-sw.cern.ch/cgi-bin/cvsweb.cgi/offline/Trigger/TrigAlgorithms/TrigT2Calo/

13-9 [T2CaloVariables]\newblock S. González, T. Hansl-Kozanecka, and M. Wielers, “Selection of high-pT electromagnetic clusters by the level-2 trigger of ATLAS,” ATLAS-TDAQ-2000-002.

13-10 [T2CaloRefSw]\newblock S. González, B. González Pineiro, and T. Shears, “First implementation of calorimeter FEX algorithms in the LVL2 reference software,” ATLAS-TDAQ-2000-020.

13-11 [T2CaloRefSwStudies]\newblock S. González and T. Shears “Further studies and optimization of the level-2 trigger electron/photon FEX algorithm,” ATLAS-TDAQ-2000-042.

13-12 [RefLVL1]ATLAS first level trigger technical design report, CERN-LHCC/98-14 (1998).

13-13 [ref:mufast]A. di Mattia et al. (Rome Level-2 Muon Trigger Group), “A Muon Trigger Algorithm for Level-2 Feature Extraction,” ATLAS-DAQ-2000-036.

13-14 [xKalman]I. Gavrilenko, “Description of Global Pattern Recognition Program (xKalman)”, ATLAS-INDET-97-165; also see: http://maupiti.lbl.gov/atlas/xkal/xkalmanpp/index.en.html.

13-15 [iPatRec]R. Clifft and A. Poppleton, IPATREC: Inner Detector Pattern-Recognition and Track-Fitting, see http://atlasinfo.cern.ch/Atlas/GROUPS/SOFTWARE/ DOCUMENTS/IPATREC/ipatrec.html.

13-16 [LArClusterRec]Put LArClusterRec reference here.

13-17 [Moore]The Moore Group, Moore – Muon OO REconstruction for ATLAS, see http://www.usatlas.bnl.gov/computing/software/moore/.

13-18 J.Baines et. al., B-Physics Event Selection for the ATLAS High Level Trigger, ATLAS Note ATL-DAQ-2000-031 (2000).

13-19 *Effects of Inner Detector Misalignment and Inefficiency on the ATLAS B-physics Trigger* by: J.Baines, B. Epp, S.George, V.M.Ghete, G.Hollyman, D.Hutchcroft, A.Nairz, S.Sivoklokov. ATL-DAQ-2001-006

13-20 *Event Filter Rate for the Ds Trigger* B. Epp, V.M.Ghete, A.Nairz. ATL-DAQ-2001-003

13-21 J.Baines et. al. Resource Estimates for the ATLAS B-physics Trigger ATLAS-COM-DAQ-2002-001

13-22 N.Konstantinidis and H.Dreverman, Fast tracking in hadron collider experiments, in Proceedings of the 7th International Workshop on Advanced Computing and Analysis Techniques in Physics Research, Amer. Inst. Phys. Conference Proceedings, Vol. 583, 2001

13-23 J. Baines et al., Pattern Recognition in the TRT for the ATLAS B-Physics Trigger, ATLAS Note ATL-DAQ-99-007 (1999).

13-24 I. Gavrilenko, Description of Global Pattern Recognition Program (XKalman), ATLAS Note ATL-INDET-97-165 (1997).

13-25 P.Billoir and S.Qian, Simultaneous Pattern Recognition and Track Fitting by the Kalman Filtering Method, Nucl. Instr. and Meth. A225 (1990) 219.

13-26

14 Overall system performance and validation

14.1 Introduction

- Definition of validation of rate capability, its context and scope.

- Summary of validation process

14.2 Integrated Prototypes

Description of the prototypes:

- HLT / PESA prototype
- integrated 10% system

14.2.1 System performance of event selection

NOTE: This section will (necessarily) be completed at a later time. The work described here is ongoing, and in some cases, not even started. The following sub-sections may be re-shuffled or modified to accommodate the evolving tests.

The High Level Trigger will select and classify events based on software largely developed in the offline environment. This approach minimizes duplication of effort and ensures consistency between the offline and the online event selections. However, given the strict performance requirements of a real-time online environment, it is essential to evaluate the performance of the HLT event selection software ("PESA software") in a realistic trigger environment.

The resource utilization characteristics of the PESA software are an important input to the models that predict overall system size and cost. For this reason, a prototyping program was developed to perform dedicated system performance measurements of the event selection software in a testbed environment.

14.2.1.1 Measurement and validation strategy

The scope of the work reported here is limited to a system with full event selection and a minimal dataflow system, providing full trigger functionality with limited performance. Such dedicated "vertical slice tests" are sufficient to test the performance of the HLT event selection in a realistic environment. Nevertheless, even in such a limited system, tests and measurements of the dataflow aspects relevant to PESA can be carried out.

An important aspect of this prototyping work is component integration. Although single components may perform very well in isolated tests, only integration with other system elements may reveal weakness not foreseen in the original design. The integration and testing work described here followed, roughly, the following steps:

5. Individual component testing and validation (addressed in Chapters 8 and 13)

6. Functional integration of relevant components (e.g., Online, Dataflow, PESA) in a small testbed, providing feedback to developers.
7. Final system validation
8. Measurement program, including event processing times, network latencies, and thread scaling.

The last three steps were carried out for a LVL2 testbed, an EF testbed, and a combined HLT testbed. The following sections summarize the outcome of this integration and measurement program.

14.2.1.2 Event selection at LVL2

Describe software components used in event selection:

- PESA Steering Controller
- Steering, Configuration. Trigger Menus
- Trigger Algorithms: T2Calo, SiTree / IDSCAN
- Data unpacking:
  - BS converters for LAr,Tile,Si / pixels
  - Event format library

Describe hardware components in testbed:

- L2PU
- ROS emulator
- L2SV
- pROS

Here will provide a table with summary performance numbers of L2 slice. The table will present overheads per component (as in ATL-DAQ-2002-012) running in one thread:

- PSC+L1Result->SG
- above+dummy algorithm requesting data containers
- above-dummy+T2CALO
- above+steering
- above+IDSCAN
- above+L2result

For above, separate network latencies and PESA execution times.

Here describe system performance results for various components in different configurations, e.g., PSC overhead, framework overhead, algorithm usage of CPU resources, number of threads, etc. Only a few results shown in a table. The rest will be in a backup document.

Also give a sense of what sort of optimization can still be done in the software/strategy so that performance can be brought to an acceptable level.

Open issues: STL., etc.?

Address robustness requirements (runs more frequently in online that in offline)?

14.2.1.3 Event selection at the Event Filter

In the Event Filter, the event selection process is carried out by the processing task (ref. to chapter 9). In the first section, the baseline choice of using the offline ATHENA framework as the processing task is described, as well as the integration of the HLTSSW and ATHENA with the Event Filter. In the second section we will describe the current testbed implementation, the measurements and the validation strategy.

14.2.1.3.1 The Event Filter Processing Task

In the Event Filter, the PESA strategy consists in using the offline reconstruction algorithms with the minimum set of adjustments required to comply with the performance goals. Although the full detector information is available in the Event Filter, it will not always be necessary to unpack the full event to reach the trigger decision. Hence, it should be possible to seed the algorithms, in particular, with the LvL2 result and the corresponding lazy data unpacking mechanism should be supported.

The baseline implementation choice for the Event Filter Processing Task is to use the offline ATHENA framework. ATHENA is the ATLAS concrete implementation of the underlying architecture GAUDI. The GAUDI architecture separates Algorithm Objects from Data Objects. The Algorithm view of the framework is shown in Figure 14-1. Each algorithm can access a set

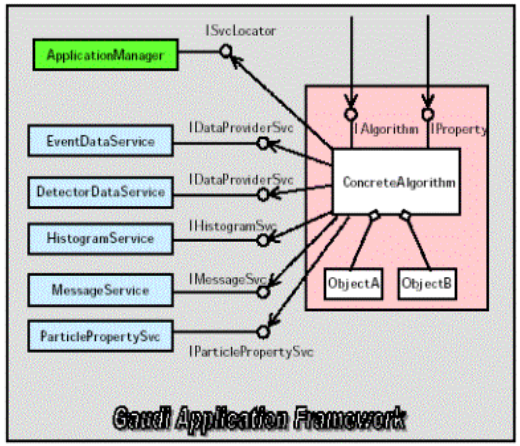


Figure 14-1 The Gaudi application framework architecture.

of services via an interface: for example the Event Data Service is accessed via the IDataProviderSvc interface. Data is exchanged between algorithms via the Transient Event Store.

The HLTSSW software is an algorithm suite steered by the Step Controller algorithm (refer to chapter XXX). While the Data Manager contributes to data preparation, all event data entities are defined in the Event Data Model. The HLTSSW is being developed in the ATHENA framework. This will facilitate the development of algorithms, the adaptation of offline algorithms, the study of the boundaries between LvL2 and EF and the physics performances studies.

As we shall see to integrate the HLTSSW and ATHENA in the Event Filter will consist in making concrete implementations of some of the ATHENA services. In the Event Filter, each Processing Task consists of a standard ATHENA process, as in the offline case, that instantiates all the necessary services and runs an "infinite" event loop.

In the following paragraphs, we will explain the event input mechanism, the production of the EFresult data fragment and how it passed to the EFdataflow, the access to the LvL2 result and the implementation of some of the services.

In the ATHENA / GAUDI concept, the infrastructure for reading from and writing to a particular persistency is provided by a conversion service (ref. Athena Developer Guide V 2.0.2). In this case the persistency is a full event in ByteStream format and the transient form are collections of Raw Data Objects while registered in the TES. The package that implements the necessary classes for a Gaudi conversion service is the ByteStreamCnvSvc. In addition, converters are responsible for converting a particular data type, in this case one for each collection type, to and from that particular persistency type.

Event Input

The service used to access the event is the ByteStreamCnvSvc. One of the elements of the service is the Input Source. Currently ATHENA requires each Event Input Source to implement the EventSelector and EventIterator interface. It is the EventSelectorByteStream that locates the requested ByteStreamInputSvc whose name is specified in the jobOptions. In case of running in the Event Filter farm, the specified input service is a class called ByteStreamEFHandlerInputSvc that specifies the event source as coming from the EFDataflow. To run in offline mode instead, reading data from a file containing events in ByteStream format, is achieved simply by requesting in the jobOptions file, a different source, the ByteStreamFileInputSvc.

When running in the Event Filter farm, the ByteStreamEFHandlerInputSvc interfaces with the EFDataflow, at initialization time, by connecting to an instance of the EFD PTclient singleton class (refer to Chapter 9). The method ByteStreamEFHandlerInputSvc::nextEvent requests a pointer to the next Byte Stream event in the EFD Shared Heap. It extracts the event size from its header and uses this information to construct an instance of the RawMemoryStorage class defined by the Event Format Library (EFL) online package. This object is in turn used to create and return an object of type RawEvent,a typedef for an Event Filter Library FullEventFragment object that is constructed from a RawMemoryStorage instance. From this point on, the treatment of data is exactly the same if running in the EF or offline.The IdentifiableContainer,used to contain the Raw Data Objects, and the corresponding converters provide a mechanism for creating the RDOs on demand.

After the HLTSSW processing is completed, its result is wrapped in an object of type EFResult which is derived from the Athena DataObject class. The ByteStreamCnvSvc is again responsible for the conversion to persistency. The list of CLIDs of objects that should be converted to persistency is declared in jobOptions. When running in the offline mode, the output service is used to simulated the events in ByteStream format; in that case the list of CLIDs of the various types of RDO collections is declared in the jobOptions. On the other hand, when running in the Event

Filter farm, one needs only to append the EFresult to the original event residing in the shared memory. Hence only the CLID of the EFresult object is declared in the jobOptions.

Output of EF selection process

After the event filtering process, an algorithm creates the EFresult object that contains a bit pattern corresponding to the result of the selection and some more detailed information about the selection, like which trigger element and menu items have been validated. The Athena ByteStreamCnvSvc calls the EFResultByteStream Converter that creates a ROD fragment which payload contains the bit pattern. The ByteStreamCnvSvc automatically takes care of collecting all existing ROD fragments and, using the Event Format Library, constructs the Full Event Fragment. The correspondence between a ROD and its corresponding ROB / ROS / SubDetector is provide at initialization. Again there is similarity between the offline case, where the aim is to produce ByteStream format event files and the case of the Event Filter test bed. Different jobOptions will select the list of CLID for the relevant RDO collections in the first case and the EFresult CLID in the latter. The same mechanism will be used to pass back to the EF additional information contained in reconstructed objects during the selection process. For example, the TES object representing an identified electron, given the corresponding converter that serializes the information and insert it in a ROD fragment, can also be included in the Event Filter "Subdetector Fragment" and will be appended to the original event.

Access to the LvL2 result

The access to the LvL2 result is a trivial matter. The converter associated to the LvL2result object will be automatically called by ATHENA and the object created in the TES after unpacking the LvL2 "Subdetector" fragment when the first access request will be issued by one the HLTSSW algorithm.

Other services

Various others ATHENA services will be interfaced to the EFsupervison, like messaging, error reporting, exception handling or to Condition Database services, etc. Again, to switch running in offline mode or running in the EF farm will consist in selecting the appropriate concrete service implementation via jobOption files.

Conclusion

We have seen that the use of the Athena software as the EF Processing Task makes it completely transparent to switch from the offline development environment to the EF farm which was one of the important requirement. Next we have to validate that this approach meets the performance requirements of the Event Filter.

14.2.1.3.2 Event Filter Prototype

To validate the choice of the ATHENA Framework for the Processing Task, an implementation of the HLTSSW running in ATHENA in the Event Filter has been prototyped in the Magni Cluster (ref. to XXX).

The strategy of validation consists in running some of the most relevant triggers in terms of rates: the electron trigger and the muon trigger. Efforts have been directed to the electron trigger first. The HLTSSW suite for electron identification is run starting with ByteStream data files corresponding to single electrons and dijet events, the main background source. The files have

been simulated offline and the result of the LvL2 selection in form of a LvL2 "Subdetector" fragment is included.

The current HLTSSW selection suite includes tracking and calorimeter reconstruction adapted from the offline electron identification software. The interface to the EF Dataflow PTclient described in the above section has been implemented. For now, the EFresult object contains a set of bits matched to the decision of the selection process: "Accept", "Reject", "ForcedAccept", "Error". No additional reconstructed objects are serialized in the current test. The implementation of the services are the following:

- The message service simply writes to a log file specific for each PTtask (name declared in jobOption file and known by the EFSupervision)
- The histogram service is interfaced to the Web based histogram service (refer to section XXX??)

Validation procedure:

1. The basic log files are monitored by the Supervision has a simple monitoring and debugging tool.
2. For timing measurement, the TrigTimeAlgs package is used. This allows to compare timing measurement done offline with the ones made in the test bed. It should be pointed out, that there is a complete decoupling between latency due to the EFdataflow and the latency of the ATHENA and HLTSSW selection process. One the pointer to the SharedHeap is passed to Athena there is no difference between that case and the offline case where the ByteStream has been read from a file and copied in the local memory. Comparing the two modes, running with equivalent processors, allows to measure the additional overhead that could arise when running, in the offline case, in an uncontrolled farm environment. The latency in the EF that may result when the request is issued for a new event can be measured with a dummy PT. This complete decoupling will not be true any more when the database access at run time will be enabled. This is not included in the current test.
3. The histogramming package is exercised. Histograms are produced by the various processing tasks and are collected by EF Supervision.
4. The Efresult will be appended to the original Event. These events are analysed and their content compared to the result of processing the same events offline.

14.2.1.4 Testing of HLT

Here will try to treat LVL2/EF as one unit. Show successful integration of LVL2 and EF in a testbed. Briefly talk about benefits of LVL2-seeded Event filter and the use of the pROS.

14.2.2 The 10% prototype

Description of the integrated 10% system

14.2.2.1 Laboratory setup

- machines, networks, OS platform(s), hardware emulators (if any)

- refer to architecture and components chapters for details

14.2.2.2 Description of the measurements

- scope of the measurement (what parameter(s) of Chapter 2, "Parameters" are we testing)

- parameter space covered

14.2.2.3 Results

- prototype results

- comparison with required performance

14.3 Functional tests and testbeam

During prototyping phases, often the performance of a system is put in foreground with respect to its stability and maintainability. Functional user requirements have in this phase of development a lower priority than the achievement of the performance requirements. This is to some extend true also for the TDAQ system, which has focused its efforts in the area of trigger rates, speed of data acquisition, etc. Nevertheless we have decided to also stress the global functionality of the TDAQ system, by carrying out a series of functional tests and exposing the system to non expert users at the ATLAS test beam sites.

Three different aspects of the functionality have been covered:

- a)Dynamic system configuration
- b)Stability in cycling through TDAQ finite states
- c)Operational monitoring and system recovery in case of errors

All these aspects have first been tested in dedicated laboratory setups and then verified in a "real" environment, during test beam data taking.

- a)A TDAQ system has to be easily reconfigurable in order to accommodate the substitution of hardware, the change of trigger conditions, etc. This means that on one side all the tools to keep the configuration parameters in a database have to be developed and on the other side that the Run Control, DataFlow and Trigger software has to be designed to be dynamically reconfigurable.

To verify the flexibility of our system the following tests have been carried out:

- -substitution of a data taking machine
- - exclusion and reinsertion of a Run Control branch
- -change of communication protocol between the ROS and the L2/EF (= change of Data Collection protocol)
- -change of run parameters.

More detailed test description and measurement results to be included here.

- b)When performing a series of measurements with different configuration options, the TDAQ system must be capable of cycling through its finite states stably. This functional requirement has been checked via automated scripts cycling repeatedly through the finite state machine.

More detailed test description and measurement results to be included here.

- c)In a distributed system such as the TDAQ it is important to constantly monitor the operation of the system. Furthermore, the fault tolerance is a fundamental aspect of its functionality. In this area several improvements are still to be achieved, but we decided to carry out a series of tests in order to assess the present performance of the system in case of errors. In particular we tried to test the fault tolerance of the system in the presence of a fatal error which prevents on or more data taking computers to continue their working.
    - Verification that all applications provide regular information on their status
    - -Failure of a SFO
    - -Failure of a EF subfarm (distributor or collector)
    - -Failure of a EF processing task
    - -Failure of a SFI
    - -Failure of a L2PU
    - -Failure of a DFM
    - -Failure of a L2SV
    - -Failure of the RoI builder
    - -Failure of a ROS
    - -Failure of a ROBIN
    - -Failure of a ROL
    -
    - -failure of online sw servers (is, mrs, ipc, ….)

More detailed test description and measurement results to be included here.

The results of the various tests will determine the summary and conclusions of this section. It is premature to indicate them now.

14.4 Paper model

14.4.1 LVL1 trigger menu

The exclusive rates for the different LVL1 trigger menu items are specified in Table 14-1. E.m./gamma (EM, the I refers to “isolated”), muon (MU), jet (J) and hadron (TAU) RoIs are distinguished, and labelled with the LVL1 energy or transverse momentum threshold. XE refers to

the LVL1 missing energy trigger. For a discussion of these menus see Chapter 4, "Physics selection strategy" (NB: 5 kHz of “Other items” are not taken into account).

Table 14-1 Exclusive rates for the LVL1 trigger menu items. For items for which two possibilities are indicated. the latter corresponds to design luminosity

LVL1 Trigger menu item	Low luminosity (kHz)	Design luminosity (kHz)
MU20	0.8	4.0
2 MU6	0.2	1.0
MU10 + EM15I	0.1	0.4
EM25I / EM30I	12.0	22.0
2 EM15I / 2 EM20I	4.0	5.0
J200 / J290	0.2	0.2
3J90 / 3J130	0.2	0.2
4J65 / 4J90	0.2	0.2
J60+XE60 / J100+XE100	0.4	0.5
TAU25+XE30 / TAU60+XE60	2.0	1.0

14.4.2 Parameters relevant for LVL2 processing

The LVL2 processing consists of several steps and after each step a decision is taken on whether data from other subdetectors within the region of interest should be requested for further analysis. In Table 14-2 the subdetectors are indicated from which data are requested in the different processing steps for the four different types of RoIs. The associated acceptance factors are also specified in the table. The data rates can be estimated using these factors and information on the sizes and the locations of the regions of interest, and on the mapping of the detector on the ROBins, in the following way: the LVL1 trigger defines a finite number of possible RoI locations. A small region in eta-phi space corresponds to each location. A hit in this region satisfying appropriate LVL1 trigger criteria generate a RoI with a location corresponding to the region. The relative RoI rate for each location is assumed to be proportional to the surface of this region, while the sum of the rates for all possible locations should be equal to the LVL1 menu RoI rate. This makes it possible to determine the rate for each possible location. In combination with the RoI sizes (see Table 14-3) and the mapping of the detector on the ROBIns the RoI data request rates for each ROBIn can be calculated. Information on the mapping can be found in ref..... (backup document on paper modelling).

In order to establish the processing resources needed for the LVL2 trigger the algorithm execution times and the overheads for sending requests and receiving data are needed. See Table 14-4 for current estimates, assuming execution on 4 GHz machines. The numbers specified include estimates of the time needed for data preparation. Furthermore for each Ethernet frame sent or received an overhead of 4 resp. 8 microseconds is taken into account. These values have been estimated from measurement results for the SFI. The processing step resulting in a decision is assumed to take 50 microseconds. Merging of event fragments into a larger fragment suitable for input in the algorithms is assumed to proceed at 160 MByte / s.

Table 14-2 Subdetector data requested by different processing steps of the LVL2 trigger for the different types of RoIs and associated acceptance factors. The acceptance factors are relative to the LVL1 RoI rate.

Type of RoI	First step	Acceptance factor	Second step	Acceptance factor	Third step
EM	E.m. calorimeter	0.19 (design lum.: 0.16)	Hadron calorimeter	0.11 (design lum.: 0.16)	TRT / SCT / Pixels
JET	E.m. and hadron calorimeters	1.0			
TAU	E.m. and hadron calorimeters	0.2	TRT / SCT / Pixels		
MUON	Muon precision and trigger detectors	0.39	SCT / Pixels	0.086	E.m. and hadron calorimeters (only for design luminosity)

Table 14-3 LVL2 RoI sizes

Type of RoI	Size in eta	Size in phi
EM	0.2	0.2
JET	0.8	0.8
TAU	0.2	0.2
MUON	~ 0.3 - 0.4 (depends on detector)	~ 0.1 - 0.4 (smallest in muon and in inner detector)

Table 14-4 Estimated execution times (in ms) of LVL2 algorithm steps on a 4 GHz processor and for low and design luminosity respectively. The estimated time needed for data preparation has been included in the RoI processing times. The algorithm execution times are the m_95 values (see chapter...)

Type of RoI or trigger	Muon detectors	Calorimeters	TRT	SCT + Pixels
EM		0.088 / 0.123 (e.m.) 0.023 / 0.032 (hadron)	8.33 / 24.56	1.36 / 3.88
JET		0.68 / 0.68		
TAU		0.044 / 0.061 (e.m.) 0.011 / 0.016 (hadron)	8.33 / 24.56	1.36 / 3.88
MUON	0.5 / 0.5	0.044 / 0.061 (e.m.) 0.011 / 0.016 (hadron)	8.33 / --	1.36 / 3.88

14.4.3 Parameters relevant for Event Builder and Event Filter

Events need to be fully built at a rate equal to the acceptance rate of the LVL2 trigger (0.6 or 1.5 kHz) and then to be analysed by the Event Filter. The Event Filter is expected to reduce the rate

by a factor of 10 (see ch....) with a typical processing time of 1 second per event, which requires a farm of at least 300 or 750 dual-CPU PCs.

14.4.4 Data rate summaries

The LVL2 system and the Event Builder both send requests for data to the ROBIns. The rate of the requests from the Event Builder is equal to the event building rate, i.e. 0.6 or 1.5 kHz. The rate of LVL2 requests per ROBIn is presented in Table 14-5. The total data volume output per ROBIn is shown in Table 14-6. See for similar tables for ROS PCs handling data from 12 ROLs (for cases where the number of ROLs is not a multiple of 12 the PC with less than 12 ROLS connected has not been taken into account for calculating the averages) Table 14-7 and Table 14-8. Network specific wrappers have not been taking into account in calculating the data volumes. For the total data volumes and for LVL2 farm sizes and number of SFIs see chapter 2.

.

Table 14-5 LVL2 request rate per ROBIn in kHz, “overall average”: averaged over all ROBIns, “maximum average”: time average for the ROBIn with the highest average number of requests

Luminosity	Muon trigger	Muon precision	E.m. calorimeter	Hadr. calorimeter	TRT	SCT	Pixels
Low (overall average)	0.02	0.04	0.42	0.27	0.03	0.11	0.13
Low (max. average)	0.04	0.06	1.19	0.40	0.04	0.15	0.20
Design (overall average)	0.10	0.22	0.61	0.31	0.01	0.27	0.34
Design (max. average)	0.20	0.30	1.75	0.45	0.02	0.37	0.49

Table 14-6 Output data volume per ROBIn in MByte/s (LVL2 data and data sent to the Event Builder), “overall average”: averaged over all ROBIns, “maximum average”: time average for the ROBIn with the highest average number of requests

Luminosity	Muon trigger	Muon precision	E.m. calorimeter	Hadron calorimeter	TRT	SCT	Pixels
Low (overall average)	0.56	0.31	0.87	0.75	0.28	0.31	0.22
Low (max. average)	0.58	0.32	1.53	0.86	0.28	0.33	0.24
Design (overall average)	1.45	0.83	1.81	1.55	1.97	2.31	1.11
Design (max. average)	1.54	0.87	2.79	1.67	1.98	2.44	1.20

Table 14-7 LVL2 request rate per ROS PC in kHz, “overall average”: averaged over all PCs, “maximum average”: time average for the PC with the highest average number of requests

Luminosity	Muon trigger	Muon precision	E.m. calorimeter	Hadr. calorimeter	TRT	SCT	Pixels
Low (overall average)	0.2	0.4	3.0	1.9	0.2	0.8	1.0
Low (max. average)	0.3	0.5	8.2	2.1	0.3	0.9	1.5
Design (overall average)	0.9	1.9	4.4	2.1	0.1	2.0	2.6
Design (max. average)	1.4	2.4	12.2	2.4	0.1	2.2	3.9

Table 14-8 Output data volume per ROS PC in MByte/s (LVL2 data and data sent to the Event Builder), “overall average”: averaged over all PCs, “maximum average”: time average for the PC with the highest average number of requests

Luminosity	Muon trigger	Muon precision	E.m. calorimeter	Hadron calorimeter	TRT	SCT	Pixels
Low (overall average)	6.8	3.8	10.8	9.2	3.4	3.8	2.8
Low (max. average)	6.9	3.9	18.2	9.9	3.4	4.0	3.0
Design (overall average)	17.6	10.3	22.2	18.9	23.8	28.2	13.7
Design (max. average)	18.2	10.6	33.3	19.6	23.8	29.2	14.6

14.4.5 The role of sequential processing

The sequential processing applied in the LVL2 trigger requires less data to be transported and less CPU time for the trigger algorithms., compared to the scenario in which all data that possibly could be needed is requested and processed. In particular analysis of the inner tracker data is less frequently needed. As shown in Table 14-4 this type of analysis is time consuming and therefore an important reduction (up to a factor of 10) is obtained by applying sequential processing, see Table 14-9. In the table also the reduction in RoI request rates and amount of data to be transferred to the LVL2 processors is specified.

Table 14-9 The effect of sequential processing on request rates, LVl2 data volume and size of the LVL2 farm.

	Low luminosity Sequential	Low luminosity Non-sequential	Design Luminosity Sequential	Design Luminosity Non-sequential
LVL2 total request rate (all ROBIns) (kHz)	387	644	572	1109
LVL2 total data volume (MByte/s)	325	467	532	1084
Number of LVL2 dual-CPU PCs	32	199	64	864

14.5 Computer model

- Discrete event simulation
- Object oriented model of system, most objects represent hardware, software or data items
- Two tools: at2sim, based on Ptolemy, and Simdaq, a dedicated C++ program
- Testbed models and models of full system.
- For the full system model the same LVL1 trigger menus as for paper model are used to generate an appropriate number and type of RoIs for each event. As in the paper model, the eta and phi coordinates of the RoIs are chosen at random from the possible eta, phi coordinates (as defined by the LVL1 trigger). The mapping of the detector on the ROBIns is the same as for the paper model. Average message rates and volumes and total CPU power utilized as obtained from the paper and the computer model of the full system therefore should be equal within the statistical errors.
- Component models described in detail in back-up document

The availability of network connections and switches with sufficient bandwidth and of a sufficient amount of computing resources in the DAQ and HLT systems does not guarantee a satisfactory system performance. The computing load should be distributed evenly over the available computing resources and congestion in switches should be prevented, as this may lead to message loss, if not enough buffer space inside the switches is available.

From the previous section and from Chapter 2 it can be concluded that the total available output bandwidth from the ROS (two Gigabit Ethernet connections per ROS PC) is considerably higher than the required bandwidth (about 28 GByte/vs. 6 GByte/s for design luminosity and a LVL1 trigger rate of 75 kHz). If single LVL2 and Event Builder switches, with the order of 200 ports and non-blocking, can be used, message loss due to buffer overflow inside the switches is unlikely. Furthermore traffic shaping as well as flow control provide the possibility to reduce the probability of message loss. Computer model results will show that building up of queues is unlikely.

The LVL2 supervisor and the DFM both can assign events to each L2PU or SFI such that the load is spread evenly. For example a simple and effective algorithm consists of the supervisor or DFM maintaining an administration of how many events are being handled by each L2PU or SFI. As the supervisor and DFM receive a message when processing is finished it is straightforward to implement this. A new event can then be assigned to the L2PU or SFI with the smallest number of events to process. Simulations of the LVL2 system have shown this to be a very effective strategy, with which high average loads of the L2PUs are possible (*reference to TPR or new results, if available*).

14.5.1 Result of testbed model

LVL2 Subsystem test, EF subsystem test, Minimal DataFlow test, larger setups..

Type of results: model and experimental results for throughput, maximum message rate, latency...

14.5.2 Results of extrapolation of testbed model and identification of problem areas

Full model

Type of results: latency, queuing in system, effectiveness of limiting output rates of ROBIns and of number of outstanding requests in L2PUs and SFIs, effect of different strategies for event assignment to L2PUs and to SFIs.

14.6 Title?

14.6.1 Technology tracking up to LHC turn-on

14.6.1.1 Network technology

14.6.1.2 Processors.

14.6.2 Survey of non-ATLAS solutions

(a reality-check on ATLAS approach?)

14.6.3 Implication of staging scenarios

Re-interpretation of performance numbers for staging scenarios

14.6.4 Areas of concern

14.7 Conclusions

14.8 References

14-1

14-2

Part 4

Organisation and Plan

15 Quality Assurance and Development Process

15.1 Quality Assurance in TDAQ

Quality assurance during the production of hardware and software systems is provided for with the adoption of a development framework for DAQ components. The development framework consists of distinct development phases. At the end of each phase a set of deliverables is provided. This framework is complemented by guidelines, checklists and standards, internal reviews, templates, development and testing tools and coding standards. Those are being adopted as common working practice and help for error removal and error prevention in the system.

A TDAQ wide body, the Connect Forum [15-1] assists in coordinating development process activities and quality assurance methodologies across Atlas TDAQ / DCS. It also provides advice, especially via the recommendations and information made available through Web pages which reflect the dynamic nature of the activity.

A common approach to the development via the use of rules, in-house standards and document templates helps in building a project culture. Those rules as well as the development phases themselves are not enforced but rather mend to be a help for developers. Emphasis on the various phases will vary and evolve with the life of the project. During event production for example, the emphasis will be put on maintenance and regular automized validation testing

A powerful release management system and a convenient working environment provide the necessary technical working basis.

15.2 The Development Process

The software development process (SDP) in Atlas TDAQ provides the structure and the sequence of activities required for development. A basic framework is provided to guide developers through the steps needed during the development of a component or a system. Continual review and modification of the SDP provides it with the flexibility to adapt to the evolution of the components and systems.

Many of the recommended approaches in the SDP are also applicable to the development of hardware components or sub-systems involving both software and hardware. The SDP consists of the following phases as shown in Figures 15-1: Brainstorming, Requirements, Architecture and Design, Implementation, Testing, Maintenance, complemented by reviews. Emphasis on the phases will evolve within time.

15.2.1 Inspection and Review

Written material including documents and code are subjected to a process of inspection and review at each step from Requirements to Implementation, in the SDP. Inspection is essentially a quality improvement process used to detect defects. The inspection process in the Atlas TDAQ project is based on Tom Gilb's Software Inspection method [15-2]. An important feature of the

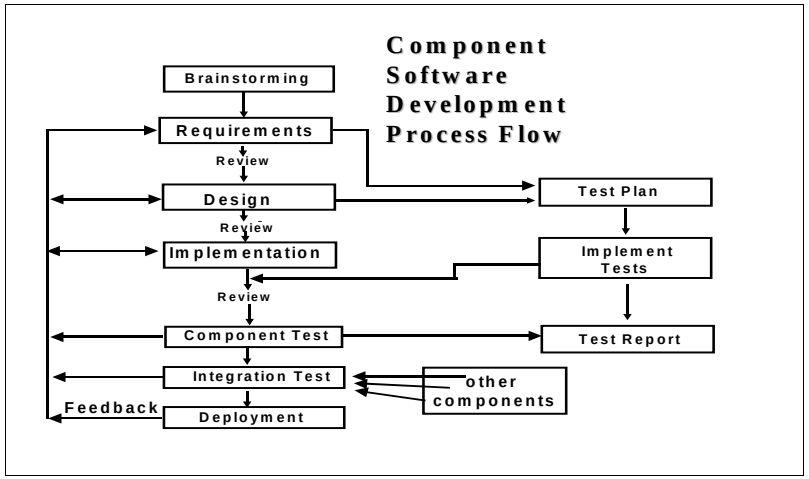


Table 15-1 Phases and flow of the Software Development Process

inspection procedure is its flexibility, allowing it to evolve as needs change during the lifetime of the project.

Overall responsibility for an inspection is taken by an inspection leader who appoints an inspection team consisting of the document author and three to five inspectors. The core of the inspection process is the checking phase where the inspectors read the document in detail, comparing it against source documents and lists of rules and standards. Defects are logged in a table, where a defect is defined as a violation of any of the standards. Emphasis is placed on finding major defects which could seriously compromise the final product. The defects are discussed at a logging meeting and their acceptance or rejection is recorded in an inspection issue log. The document author edits the document according to the log making an explanatory note if an issue is rejected. Feedback is also obtained on how the inspection procedure itself may be improved.

The principal aim of inspection is to detect and correct major defects in a product. An additional benefit is the possibility to prevent defects in future products by learning from the defects found during inspection procedures. Inspection also provides on-the-job education to people new to a project and generally improves the project's working culture.

A number of web pages have been produced which provide supporting material for inspections such as instructions for inspectors and log file templates[15-3].

15.2.2 Experience

The Software Development Process provides a disciplined approach to producing, testing and maintaining the various systems required by the ATLAS TDAQ project. It helps to ensure the production of high quality software and hardware which meets the requirements within a predictable schedule.

However, one of the key differences in adopting the SDP in an HEP as opposed to industrial environment is that its application cannot be enforced. Furthermore, the use of such a process may appear too rigid to physicists not accustomed to working in a strong management framework. Nonetheless, the working culture can be changed by increasing awareness of the benefits of the SDP through training, for example involving new group members in inspections, and ensuring that the SDP itself is sufficiently flexible to evolve with the changing needs of an HEP experiment. This approach is working. The SDP as outlined in this section has already been adopted by a number of the sub-systems in the ATLAS TDAQ project with positive outcomes [refs].

15.2.3 The Development Phases

15.2.3.1 Requirements

The Requirements phase for a particular sub-system or component consists of gathering the requirements and then documenting them. Several documents have been produced to aid and control these activities, based on the early experience of some of the sub-systems. The whole process of working group setup, requirements collection, feedback & review is described [15-4]. Another document [15-5] sets out the principles governing the requirements gathering and documentation processes, stressing the importance of, for example, documentation, evolutionary development, communication, and collective ownership of the requirements specification.

The actual process of establishing the requirements for a sub-system or component is aided by a collection of "hints" [15-6], and reinforced by a set of 22 rules [15-7] for the requirements document itself, for which a template [15-8] has been provided in each of the supported documentation formats.

15.2.3.2 Architecture and Design

The Architectural Analysis and Design Phase of the SDP follows the Requirements phase and takes as its starting points the User Requirements & Use Cases together with accompanying documents. This phase has sometimes been referred to as "high-level system design". A system's architecture is the highest level concept of that system in its environment. It refers to the organization or structure of significant components interacting through interfaces, those components being composed of successively smaller components and interfaces. A design presents a model which is an abstraction of the system to be designed. The step from a real world system to abstraction is analysis. A 'Howto' note [15-9] has been produced describing the overall process.

For this phase, we are largely following the approach of the Rational Unified Process (RUP), which contains descriptions of concepts, artifacts, guidelines, examples and templates. In particular, we have highlighted the RUP descriptions of architectural analysis and design concepts [15-10] and guidelines for producing software architecture and design documents[15-11].

We have adapted the RUP template for architecture and design documents by including explanations and making it available in supported formats[15-12]. The recommended notation is the Unified Modelling Language (UML), and the design is presented in the template as a set of UML-style views. We have also prepared recipes for producing appropriate diagrams and incorporating them into documents.

15.2.3.3 Implementation

The Implementation Phase of the SDP is largely concerned with writing and checking code and producing and debugging hardware components. At the end of the implementation phase an Inspection is performed.

ATLAS C++ coding conventions[15-13] are being applied to newly written code and being introduced for existing code still in evolution. In the case of Java we await the outcome of the Atlas investigation of coding conventions. DCS will follow the coding standards provided by the JCOP Framework for PVSS [15-14].

Guidelines [15-15] have been provided for multi-user multi-platform scripting, as well as many explanations and examples in unix-scripting[15-16].

Experience has been gathered with a number of software tools and recommendations have been made in the areas of design and documentation [15-17], code checking[15-18], and source code management[15-19]. No standards have been identified for the hardware so far.

15.2.3.4 Component Testing and Integration Testing

Testing occurs during the entire life-time of a component, group of components or entire system. Referring to figure [SDP], the initial test plan is written during the requirements and design phases of the component, so as not to be biassed by the implementation. Since testing is likely to be an iterative process the test plan is written with re-use in mind. Once implementation is complete and passes relevant checking tools the component undergoes unit testing to verify its functionality. Compatibility with other components is verified with integration tests. Several types of tests can be envisaged for both individual components and groups of components. These include functionality, scalability, performance, fault tolerance and regression tests.

A test report is written once each test is complete. To aid the testing procedure, templates [15-20] are provided for both the test plan and test report in each of the supported documentation formats. More detailed descriptions of the types of test, hints on testing and recommended testing tools are also provided [15-21]. Testing is repeated at many points during the life-time of a component, for example at each new release of the component software or after a period of inactivity (system shutdown). Automatic testing and diagnostic procedures to verify the component before use greatly improve efficiency.

15.2.3.5 Maintenance

As with testing, maintenance occurs during the entire life-time of a component. Several types of maintenance can be envisaged. Corrective maintenance involves the fixing of bugs. Adaptive maintenance involves alterations to support changes in the technical environment. Preventative maintenance entails the restructuring and rewriting of code or modification of hardware for future ease of maintenance. Maintenance is closely coupled to regression testing which should occur each time a maintenance action has been completed to verify that the detected problems have been fixed and new defects have not been introduced. Significant changes to the functionality of the component such as the addition of large numbers of new requirements should involve a full re-iteration of the SDP cycle.

15.2.4 The Development Environment

Regular releases of the sub-system software to be used in test beam operation, system integration and large scale tests is being complemented by nightly builds and automated tests to ensure early problem finding of newly developed or enhanced products. The use of a source code management system and of the standard release building tool CMT [15-19] allows for the building of common releases of the TDAQ system. These releases are available for the platforms used in Atlas TDAQ which are currently a number of Linux versions and for some sub-systems LynxOS and SunOS. Build policies of different sub-system like the use of compiler versions and platforms are coordinated.

Development tools like design tools, memory leak checking tools, automatic document production tools and code checking tools are vital elements of the development environment.

No standards have been identified for the hardware so far.

15.3 Quality Assurance During Deployment

15.3.1 Quality Assurance of operations during data taking times

The quality of the DAQ system must be assured when it is in use during the setup and installation phase of the Atlas data acquisition together with the detectors. Correct and smooth data taking shall be aimed for during calibration and physics event production.

Quality assurance is achieved by prevention, monitoring and fault tolerance.

- prevention: this includes training, appropriate documentation, a well defined development process, proper management of computing infrastructure (computer farms, readout electronics and networks), tracing of hardware and software changes, regular testing of components.
- monitoring: special tasks to monitor proper functioning of equipment and data integrity. These may run as special processes or be part of the TDAQ applications. Anomalies are reported, analysed by human / artificial intelligence and appropriate recovery action is inititated. This may include running special diagnostic code, replacement of faulty equipment, rebooting of processors, restarting of applications, re-establishing network connections, re-configuration to continue with a possibly reduced system. Incomplete or corrupted data should be marked in the event data stream and possibly recorded in the conditions database. Physics monitoring may lead to a change of run with different trigger conditions and event selection algorithms.

fault tolerance: built into the system from the start and using an efficient error reporting, analysis and recovery system this provides the basis (cf. chap.6 for details). Some redundancy to reduce possible single point of failures is foreseen where affordable (cf. chap. 6).

During the life of the experiment small or major pieces of hardware or software will need to be replaced with more modern technology ones. The component structure with the well defined functionality of each component and well defined interfaces allowing for black-box testing according to those functionality specifications will allow to incorporate smoothly new parts into a running system, in particular also when staging of the system is required.

15.4 References

15-1 http://atlas-connect-forum.web.cern.ch/Atlas-connect-forum/

15-2 reference to Tom Gilb's inspection method

15-3 TDAQ inspection web pages

15-4 Practical Steps towards an Atlas TDAQ Requirements document.

15-5 Requirements gathering and documentation "principles".

15-6 Hints on how to establish requirements.

15-7 Requirements Document Rules ATLAS DAQ 'in-house' rules for Requirements documents. ID: ATD-R-R1.

15-8 Requirements Document Template for Systems and Components, Software and Hardware.

15-9 How-to for Design

15-10 RUP URL for concepts.

15-11 RUP URL for guidelines.

15-12 Template for Software Architecture Document.

15-13 ATLAS C++ Coding Standard Specification The coding conventions.

15-14 JCOP Framework for PVSS.

15-15 Multi-user multi-platform scripting guidelines.

15-16 Unix-scripting examples and explanations.

15-17 Doxygen, Visual Thought, Together, DOC++, Source Navigator

15-18 RuleChecker

15-19 CVS, SRT, CMT

15-20 reference to templates on web page

15-21 reference to testing web page

open points:

- HW inventory and information logging

16 Costing

16.1 Initial system

16.2 Final system

16.3 Deferral plan

16.4 References

16-1

16-2

17 Organization and resources

Should the geographical, racks, power supplies, and cooling issues be addresses in this chapter or in the system component ones?

17.1 ...

17.2 References

17-1

17-2

ATLAS
High-Level Triggers, DAQ and DCS
Technical Design Report
30 June 2003

18 Work-plan

Post TDR. Need to change the title of the chapter I think

18.1 Schedule

This section will present the overall schedule for the HLT / DAQ up to LHC turn-on in 2007. The principal milestones coming from: detector needs for TDAQ in installation, TDAQ component production (in the case of custom components like the ROBin & the RoIB), component purchasing & associated tendering, component testing time etc. Probably require a MSproject style diagram plus associated details.

18.2 Commissioning

Description of first ideas for the commissioning process of the TDAQ as well as the steps in the detector commissioning which need TDAQ components.

18.2.1 TDAQ

How are the various elements of the system will be commissioned. What other elements will be required for this

18.2.2 Tools for detectors

More details on which elements of TDAQ will be required by the detectors, and when, to progress with their commissioning plans

18.3 Workplan up to June 2005

Here we should discuss in some detail the work needed to be done in the coming 2 years to arrive at decisions and final choices in the various areas of the system. The year 2003 / 4 should be presented in as much detail as possible, enumerating specific dates where possible. The work for the following year will probably contain somewhat less detail.

18.4 References

18-1

18-2

This document has been prepared with Release 5.5 of the Adobe FrameMaker® Technical Publishing System using the Technical Design Report template prepared by Mario Ruggier of the Information and Programming Techniques Group, ECP Division, CERN, according to requirements from the ATLAS collaboration.

To facilitate multiple author editing and electronic distribution of documents, only widely available fonts have been used. The principal ones are:

Running text:	Palatino 10.5 point on 13 point line spacing
Chapter headings:	Helvetica Bold 18 point
2nd, 3rd and 4th level headings:	Helvetica Bold 14, 12 and 10 point respectively
Figure and table captions:	Helvetica 9 point