EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

CERN/LHCC 95-47
LCRB Status Report/RD13
1 August 1995

Status Report of
A SCALABLE DATA TAKING SYSTEM AT A TEST BEAM FOR LHC

C.P. Bee[1], R. Jones, S. Eshghi[2], S. Kolos[3], C. Maidantchik[4], L. Mapelli[5], G. Mornacchi[6], M. Niculescu[9], A. Patel[7], D. Prigent, R. Spiwoks[8], I. Soloviev[3].
CERN, Geneva, Switzerland

M. Caprini[9], P.Y. Duval, F. Etienne, D. Ferrato, A. Le Van Suu, Z. Qian, S. Tisserant, F. Touchard
Centre de Physique des Particules de Marseille, IN2P3, France

I. Gaponenko, Y. Merzliakov
Budker Institute of Nuclear Physics, Novosibirsk, Russia

F. Harris, S. Hunt
Nuclear Physics Laboratory, Oxford University, Oxford, United Kingdom

G. Ambrosini[10], R. Ferrari, G. Fumagalli, G. Polesello
Dipartimento di Fisica dell'Universita' e Sezione INFN di Pavia, Italy

1. Now at University of Zurich, Switzerland.
2. Now at University of Utrecht, The Netherlands.
3. On leave from the Petersburg Nuclear Physics Institue, St. Petersburg, Russia
4. Also with Federal University of Rio de Janeiro, Brazil.
5. Spokesperson.
6. Contact Person.
7. On leave from School of Computing and Information Systems, University of Sunderland, UK.
8. Also at the University of Dortmund, Germany.
9. On leave from the Institute of Atomic Physics, Bucharest, Romania.
10. Now at University of Bern, Switzerland.

1 Introduction

Approved originally in April 1991 for the study of an LHC oriented data taking system [1], the RD13 project has now completed the fourth phase of its development according to the plan proposed and approved in June 1994 [3]. We recall the main objectives of the project:

- The construction of a Data Acquisition (DAQ) framework which satisfies requirements of scalability, modularity and openness and serving as an ideal environment for:
- R&D studies of full architecture solutions (modelling) and of their building blocks, such as event building and processing systems for high level triggering.
- The use of real-time Unix operating systems as one such building block, especially when coupled to RISC technology.
- The study of integrating commercial software products as cost effective solutions for the development of big complex systems.
- The exploitation of software engineering techniques to establish their suitability for DAQ software design and developments.

Such investigations are more effective if performed in a realistic environment. It is, therefore, a fundamental requirement of RD13 that a fully operational DAQ system, based on the elements outlined above, be developed in versions of increasing performance and functionality and used as a data taking system of LHC detector prototypes.

This report consists of two parts:

- Part I gives an account of the activities performed and the results obtained. It is structured in 4 main chapters:

1. The RD13 DAQ system: where we give an overview of the evolution and the use of the RD13 DAQ in test beam runs.
2. Laboratory developments: it describes the major activities studying DAQ hardware and software components.
3. Architecture Studies: a combination of modelling and prototyping work has been performed in the context of the event builder, level-2, level-3 sub-farm and a generic architecture for the DAQ of an LHC-like experiment.
4. Object Oriented Software Development: the evaluation of software technology has mainly been focused on object oriented systems. Both on the development side, methodologies and associated CASE tools, and in the area of data bases.

- Part II presents an assessment of the RD13 project achievements and ideas on how the results could be carried over to and exploited in an LHC experiment.

PART I

Activities and results

2 The RD13 DAQ system

2.1 Working Principles

In order to put into context the following sections, we briefly remind the basic working principles of the RD13 DAQ system [3]:

- The system is organised in terms of a front-end component, dealing with the read-out and data flow, and a back-end component, responsible for development, data bases, user interface, monitoring and controls.
- Data bases describe the hardware, the software and the read-out configurations of the system.
- The data flow is driven by a flexible protocol covering both the main (from detector to recording medium) and the analysis (sampling for monitoring purposes) data streams.
- The main data flow can be configured either to directly read-out a detector (single detector mode) or to merge data produced by detector specific DAQ systems (multi-detector mode). Multi-detector mode is supported by an interface (hardware plus portable software) and by an optional RD13 local DAQ skeleton.
- A hierarchical control system supports, via data base descriptions, various configurations.
- A number of ancillary sub-systems, e.g. message distribution and browsing, status display etc. complete the DAQ.

2.2 DAQ Evolution

The RD13 DAQ system has evolved to cope with the demanding requirements of a configurable multi-detector setup. This has happened through a number of test beam runs with different detector R&D's in summer 94, culminating with a combined run of several ATLAS sub-detector prototypes in September 94. This version of the RD13 DAQ system was left in the H8 test beam area and it is currently used by ATLAS sub-detectors during 1995. Further DAQ developments are introduced in the test beam system when ready for production and when the need arises.

2.2.1 New Features

Since June 1994, a number of functions have been completed and new features have been added, namely:

- The support for scaling in the number of detectors (data sources) has been completed to include: merging of data coming from local detector acquisitions (event building), configuration description in terms of data bases, support for the control of multiple acquisition systems by the run control module.

- The system, as mentioned above, is configurable via data bases and supports (both in the pure acquisition and in the run control) stand-alone (one detector only taking data without a data merging stage) and multi-detector setups.
- The Fast Data Link (FDL) hardware system [4] from CES was integrated, as an optional alternative to the VIC8251 [5] based data merging.
- The recording sub-system was enhanced by two new features:

1. support for labelled tapes (when recording data locally in the test beam area), and
2. integration of the central data recording system (recording events directly to the CERN computer centre) developed by the CN division [6].

- The run control system has been enhanced, in addition to the developments required by the multi-detector setup, by a centralised user interface system and a process control package.
- Object oriented technology has been extensively used in the area of data bases:

1. Two new data bases, the first to manage status and error messages [TN137] and the second to maintain run bookkeeping information [TN141], have been developed based on the OMW [45] CASE tool.
2. Editors for existing data bases, e.g. the hardware description data base [TN155], have also been developed using OMW.
3. The data bases originally implemented with QUID [16], [3] were re-implemented with the ITASCA OODBMS [47].

- New graphical products [7] were used to improve the status display and the event dump programs.

2.2.2 Test Beam Runs

The RD13 DAQ system has been in continuous use, since June 1994, by ATLAS detector prototypes for their runs in the H8 area. The setups have included different configurations both in single and multi-detector mode.

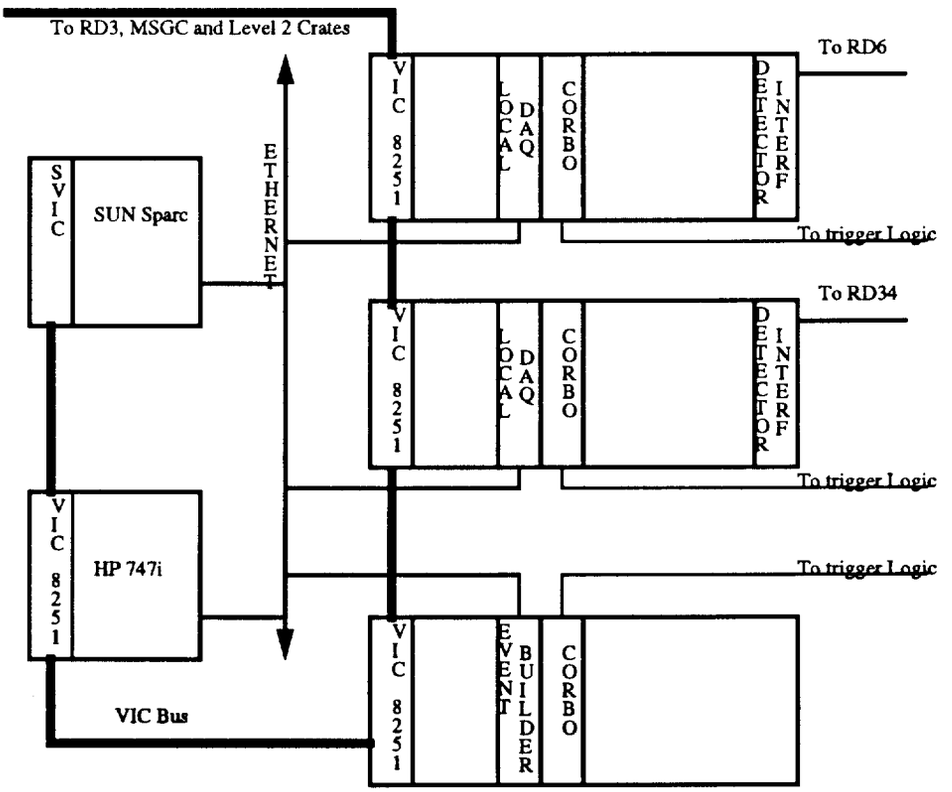
- The ATLAS hadron calorimeter (Tile calorimeter) [9] in June 1994
- The ATLAS transition radiation tracker (TRT) [8], silicon detector [10] and second level trigger prototype [11] in June-July 94.
- The Tile calorimeter in July-August 94.
- The TRT in August- September 94.
- The combined ATLAS run, including the Tile calorimeter, Liquid Argon (Lar) calorimeter, TRT and micro-strip gas chamber (MSGC) detectors, in September 1994.
- The Tile calorimeter in May-July 1995.
- The TRT in July-August 1995.

The first series of test beam runs (in the period June-July 1994) was used, by RD13, to validate in real life the multi-detector system built and tested in the laboratory as wall as a means to produce performance figures in real-life conditions. Of course this was done without disturbance to the detector data taking.

We will not go into any details of the different detector setups. Figure 1 shows, as an example, a sketch of the 1994 ATLAS combined test beam run. We wish nevertheless to highlight a few general points on the system. The system was ready on schedule, fully functional and behaved reliably. The performance was more than adequate, in certain cases was even an order of magnitude higher than required. The process of integrating into the multi-detector configuration different (and fully func-

tional) DAQ systems, such as those of the LAr calorimeter and the MSGC detector, proved to be simple and smooth.

Figure 1 Example Multi-detector configuration



3 Laboratory developments

3.1 Platform Independence

The activities in the area of platform independence and real-time operating systems have continued in a number of directions: the completion of the DAQ porting to platforms running LynxOS, the port to the Solaris operating system and the use of VME boards running workstation-like operating systems.

The RAID 8235 [13] card with EP/LX [22] is a stable product, no new releases of EP/LX have been installed. It is routinely used for production in the test beam, we foresee it can remain one of our main production platforms for the next 2 to 3 years.

The full RD13 DAQ port to LynxOS [23] for the Motorola 680x0 CES boards has been completed. The FIC8234 [26] and the VCC2117 [28] boards, the first is a generic VME based processor board and the second combines CAMAC and VIC interfaces, can today be used as local detector DAQ processors. The FIC8234 may also be used to run the full RD13 DAQ. This exercise included the port of all the commercial software products which are used by the RD13 DAQ, including ISIS, QUID

and the ITASCA client component. Simple recompilation was not enough and some substantial effort was needed, thus reinforcing our initial impression that the UNIX compatibility of LynxOS is not at the same level as that of EP/LX. On the pure real-time aspects of the system, the opinions expressed in [3] still hold.

In spite of the above reservations and given the very unlikely availability of EP/LX on platforms different from the RAID 8235, the porting exercise to LynxOS is a basic step in the direction of upgrading the system to use modern processor hardware (such as boards based on the PowerPC chip [29]).

The porting to the Solaris [30] operating systems was performed, thus widening the spectrum of platforms usable for the RD13 DAQ. The conversion of the basic RD13 DAQ software was done with minimal work, the temporary unavailability of some of the commercial products prevented us from running some of the DAQ components in native mode (but they can still run in SUNOS 4.x emulation mode).

VME boards based on processors running "standard" workstation UNIX systems, such as HP-UX or Solaris, are of interest for two main reasons:

1. Economy: they combine workstation functionality and VME interface more cost effectively than a full workstation with a VME interface. They are a convenient alternative as back-end "workstations".
2. Real-Time aspects: commercial software products are increasingly pervasive in the DAQ. They are readily available to the UNIX workstation market, but the same is not necessarily true for niche market operating systems such as LynxOS or real-time kernels. Hence the need for substantial (and expensive) porting efforts. The possibility of using for many of the front-end functions a workstation-compatible VME board would greatly reduce DAQ maintenance and porting costs.

We have explored in these directions by means of two VME boards: the HP743rt [31] (running HP-UX) and the Themis SPARC MP [32] (running Solaris).

Their integration as back-end workstations with VME access was simple since our system already runs on both HP-UX and Solaris; the only development involved was the implementation of our VME access library for the Themis card.

The real-time aspect is more complex, and it depends on a re-assessment of both the real-time requirements for a next generation of DAQ systems and medium term future for commercial and standard operating systems. Some initial exploration is being done based on the combination of a Themis card with the Solaris operating system. This latter promises to have real-time capability.

RD13 has been working in the area of real-time, UNIX-like, operating systems throughout its entire life-time. Bearing in mind the objectives and the time scale of the project, we can draw the following conclusions:

- The initial decision of using a real-time UNIX system was important to the success of the project. We could benefit from the best of two worlds: few microseconds to few 10s of microseconds for the most important system operations (such as interrupt response, context switching, synchronisation, etc.) while maintaining a very high degree of compatibility with workstation operating systems.

- The choice of LynxOS, first in its EP/LX incarnation and than in its "pure" form, showed to be the correct one. When we started the real-time UNIX market was in its infancy, today LynxOS has emerged as the most widespread product.
- In a production context, such as a test beam environment, an approach based on LynxOS will maintain its validity in the medium term (2-4 years).
- In the long term, in particular with respect to the LHC time scale, a POSIX-like interface [24], [25] seems today the best effort towards standardisation. Nevertheless we should keep an open mind. Consideration should be given to other operating system environments. Such as WindowsNT whose penetration might grow beyond the personal computer application, and operating systems based on micro-kernel technology, which possess an attractive technical potential.

3.2 Fast Data Link

The Fast Data Link (FDL) 8050 [4] hardware system was, following some initial evaluation work [TN138], integrated as an optional alternative to the VIC8251 based data merging. Performance measurements were made in order to assess the potential of the FDL system to provide high speed transfers in the DAQ environment.

3.2.1 Event Builder Application

The event builder merges the different pieces of data corresponding to the various detectors (sub-events) and stores them in a memory buffer. The detector raw data are initially stored in memory modules of different VME crates (local DAQ buffers), supervised by a standard buffer manager. An FDL 8050 module is associated to each local DAQ and it is chained to another FDL 8050 module playing the role of the data merging hardware.

For each event to be built, the event builder module collects information of where the sub-events are, issues an FDL transfer request per sub-event and then waits for the interrupts signalling the completion of the transfers. The above functions are supported by a software library [TN142] built on top of the CES provided firmware.

The FDL system supports a rich set of transfer modes [4], the VME block mode[1] is the one which suits our application. Parametrisation, in terms of VME transfer mode (D32 or D64, block size[2] from 4 to 1024 bytes) and flow control on the FDL line, is possible. The selection of the parameters depends on the hardware being used (VME modules may or may not support D64) and on the size of the sub-events; results from measurements (see section 3.2.2) can be used to select appropriate values for the above parameters.

1. This is the simplest type of transfer mode: the FDL system is required to copy a block of VME memory from a VME crate to another piece of memory.

2. The VME block size defines the maximum size (in bytes) for an atomic transfer on VME, the larger the block size, the faster (potentially) is the transfer on VME.

3.2.2 FDL Performance Measurements

The performance of the FDL 8050 system has been studied with the objective of understanding both its general behaviour and its use in the context of the DAQ event builder application.

The results from the analysis of the basic FDL behaviour are reported in [TN138]. Performance in an event builder-like environment has been studied both by means of a skeleton data transfer program, reproducing the data merging functionality of the event builder, and by using the full DAQ system configured to run with the FDL. In both cases the VME block transfer mode is used.

Figure 2 shows results obtained with the skeleton program running on a RAID 8235 and moving data either between RAID 8235 memories (for the D32 transfer mode) or between FIC 8234 memories (for the D64 transfer mode). We have measured the time between issuing a transfer request and its completion; this time includes both the actual FDL transfer time and the entire overhead involved by requesting a transfer and receiving the completion confirmation. Measurements were made with different values for the transfer parameters: D32 or D64 mode, various VME block sizes, enabling or disabling flow control on FDL[1] and of course with different "event" (amount of data to be transferred by the FDL system) sizes. In Figure 2 we show three parameter settings with the largest allowed block size (1024 bytes), using D32 or D64 mode and with or without flow control. We also show a performance curve for D32 mode, block size of 64 and flow control; a realistic parameter setting for the DAQ system and the one used by default with the full RD13 DAQ. Figure 2 suggests a number of observations:

- For small messages (up to a few Kbytes) transfer rates of a few Mbytes/second are obtained, without any dramatic effect when varying parameters (e.g. when comparing D32 and D64 we do not get an improvement close to a factor of two until messages of over a few 10s Kbytes are transferred). This indicates that a substantial overhead is involved: part of it is software related (the program is going through a library and needs to initialise a number of FDL registers) but a sizeable fraction is due to the hardware. As a confirmation of the above, when we measure the transfer of 4 bytes, in either D32 or D64 mode, we get a transfer time (which we estimate to nearly all overhead) of about 500 microseconds.
- While flow control has little effect for messages up to 10Kbytes, it does influence performance for larger message sizes. A difference close to 50% in favour of the case without flow control is seen for messages larger than 100Kbytes. Disabling flow control seems to be the means to reach high speeds, however this may lead to reliability problems.

1. Flow control is related to when an FDL module decides to transmit the next packet towards a destination; with flow control enabled the source FDL module waits for an acknowledge on the previously sent packet. When flow control is disabled, the acknowledging mechanism is not used. This latter option leads to the highest transfer rates, in particular in D64 mode, but its use is discouraged by the manufacturer which suggests reserving its use for broadcast operations only.

- The VME block size also has an effect on the transfer speed, we show in the following table the transfer rates for different block sizes in D32 and D64 modes without flow control. While D32

Table 1: Mbytes/sec as a function of VME block size for D32 and D64 transfers.

VME Block Size	D32	D64
64	17.5	23.3
128	17.8	30.3
512	20.6	34.7
1024	21.9	39.2

mode is not too sensitive to block sizes, the opposite is true for D64. The choice of the block size is therefore another important parameter to be tuned to the application's needs.

As a final remark, the maximum performance is obtained for messages of several 100's Kbytes, in D64 mode, with a 1024 VME block size and without flow control. In these conditions we measure a speed in excess of 39 Mbytes/second.

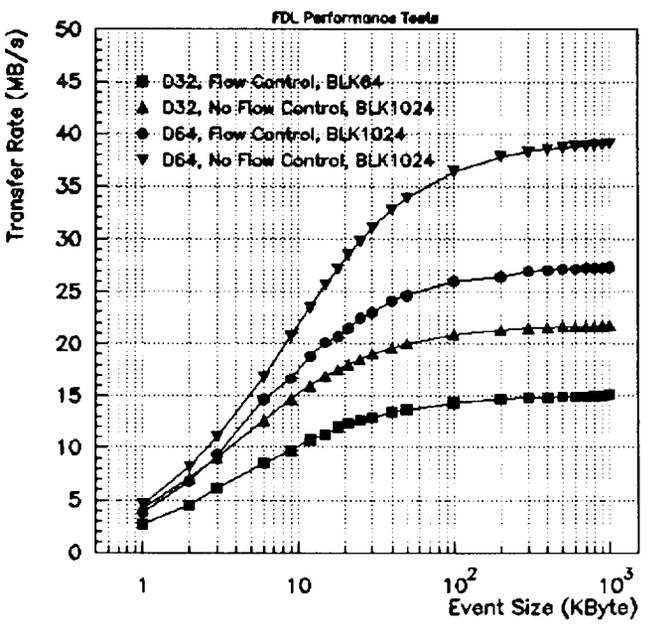


Figure 2 FDL Performance Tests

Laboratory measurements for transfers using the full DAQ system are shown in Figure 3 where the DAQ is configured to read out one sub-detector using the FDL system.

For the lowest curve, the detector memory resides in a RAID 8235 (sub-detector memory) and the event is built in the memory of another RAID processor. The results shown refer to a selection of the parameters corresponding to D32 mode, flow control enabled and VME block size of 64 bytes (as for the set of measurements marked with a square in Figure 2). As far as the handling of the FDL transfer is concerned, the main difference with respect to the case of Figure 2, is the fact that now the DAQ event builder receives interrupts signalling the end of a transfer (as opposed to polling a status bit). The use of the interrupt is justified by the fact that the RAID processor is shared by the event builder with other activities (e.g. recording). At large message sizes we find the same results as with the skeleton program, the same is not true for small message sizes (up to about 10Kbytes). The additional overhead experienced, because of the DAQ inter-process synchronisation and the interrupt handling, is responsible for this discrepancy.

The two other curves refer to setups potentially providing the maximum performance, but whose reliability in the context of the full DAQ system has yet to be studied. They are related to the use of FIC 8234 as memory modules, without flow control and with a VME block size of 1024: one refers to a D32 transfer mode, the other to the D64 case.

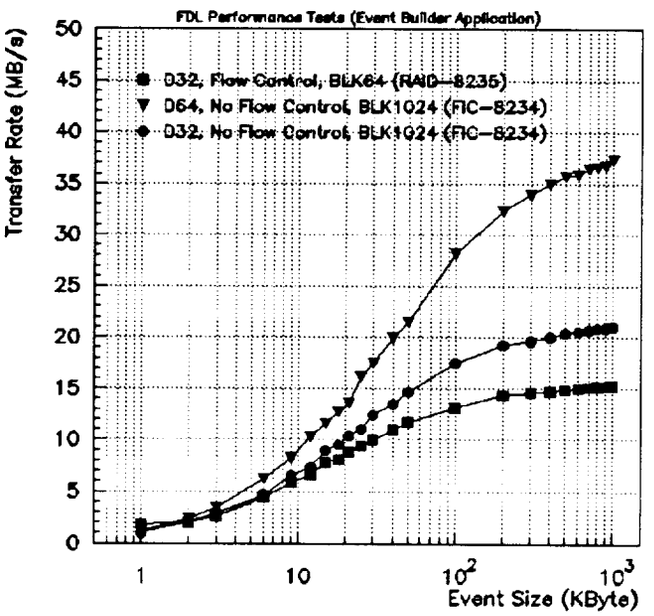


Figure 3 FDL performance (RD13 Event Builder Application)

Performance wise the use of the FDL system is not cost-effective in the case of small sub-events (few Kbytes) and low rates (few hundreds of events per burst) typical of today's test beam applications. The basic overheads involved are relatively important. The global overhead in excess of 500 micro-

seconds we have measured can be probably improved by some optimization in software. However tests made by ECP/ESS group [33] also show the existence of a substantial overhead in the FDL system itself and they reach similar conclusions.

The FDL system, however, is an interesting option and justifies further investigation for a number of reasons.

- When several sub-detectors are being read-out it provides an aggregate bandwidth higher than the VIC, multiple event transfers could be initiated concurrently and of course the FDL option is a valuable solution for large sub-events. For these reasons we will provide optional FDL support in the current DAQ system, with the possibility of optimising the choice of parameters as a function of the detector configuration.
- We have replaced the data transfer algorithm with FDL driven transfers. However the migration of more "event builder" functionality into the FDL can be envisaged. The separation in the FDL firmware between the protocol handling and the data transfer handling makes possible to implement the whole data merging algorithm directly in the FDL firmware.
- The FDL system has potential use elsewhere in a DAQ system; for example as a data distributor in a sub-farm.

3.3 Run Control and DAQ User Interface

Progress in he control and run time management of the RD13 DAQ was achieved in several areas: an improved control scheme, to cope with multi-detector DAQ requirements, a generic DAQ user interface and a distributed process management package.

3.3.1 Run Control Library and Scheme Evolution

The step to support scalability in terms of detectors and the heavy use of the system in the test beam lead to additional requirements for the run control system. This resulted in the addition of new features to both the basic run control (RCL) library [TN94] and the control scheme.

Detector selection. The possibility to dynamically (i.e. at start of run) select/remove detectors in/from the event builder read-out had to be complemented by a similar function related to detector control modules.

Detector Control. The integration of several detector DAQ systems, some of which were developed independently from the RD13 DAQ, resulted in the extension of the run control scheme and features. The notion of a local, detector specific, control module was introduced as an integral part of the hierarchical control tree [TN94]. Additional control task types (e.g. processes to be activated at start or end of run) were introduced to cope with specific detector actions to be executed upon the occurrence of a control event (e.g. an end of run).

Integration methods. The integration of new services (e.g. DMW, tape labelling) and features (local controllers) resulted in a further assessment of integration problems. This lead to a useful review of the internal organisation of the run control system.

Coherency Maintenance. The introduction of problem diagnosis and recovery in the run control system is still too ambitious a step for the current system. Nevertheless some degree of "coherency maintenance" was introduced (e.g. to recover from the situation where a detector control sub-system makes a state change inconsistent with the state of other detectors.)

Special Messages. Some new functions were added to the run control library to enable control modules to send each other application defined messages and use private (one-to-one or one-to-many) information exchange protocols. This facility is used for instance by the run control to inform the DAQ main window (DMW) of local acquisition process creation or death.

Debugging Facilities.The need was felt for a facility to exercise the run control system in stand alone mode (i.e. without the presence of the whole DAQ). Such a tool was developed and used for debugging purposes.

3.3.2 Run Control User Interface

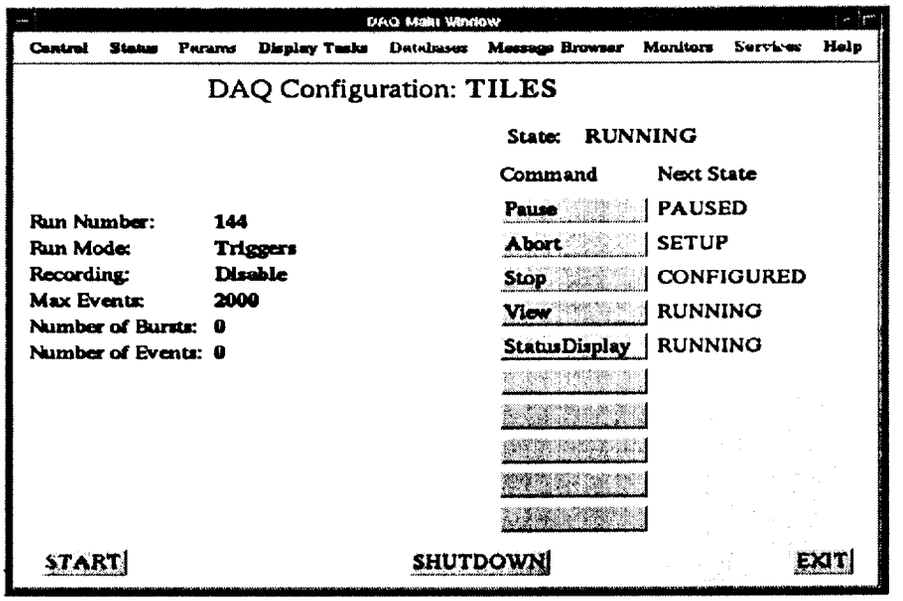
The issue of a generic, non-expert oriented, user interface to the control of the RD13 DAQ was tackled with the primary objectives of:

- unifying the access to the interactive interfaces of different DAQ components (e.g. data bases),
- running the DAQ system from a simple (e.g. hiding the complex, distributed nature of the system), yet fully functional, interactive program and
- providing a dynamic, synthetic view of the status of the DAQ

The DAQ Main Window (DMW) component [TN152] was thus defined and developed. It is currently in use in the RD13 laboratory to provide:

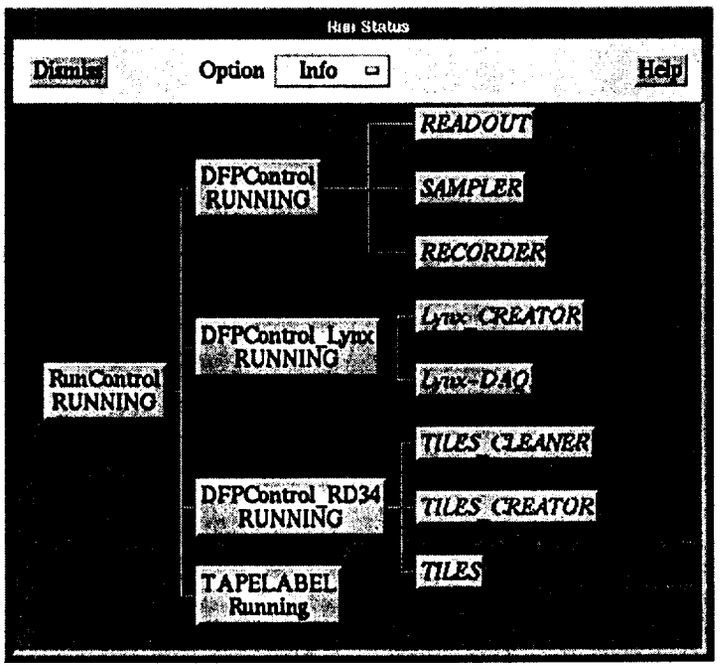
- selection of system configurations (data bases),
- initial DAQ startup, DAQ state change (e.g. run) push-buttons (Figure 4.),
- run state information (updated dynamically, Figure 5.) and
- access to all DAQ utilities (e.g. status display, monitoring tasks, etc.) via pull-down menus.

Figure 4. DMW main window



DMW integrates several commercial packages including Motif graphics, ISIS and data bases. This resulted in a number of integration problems since different tools promote different styles for steering and organising programs. In the end a compromise was found limiting the use of a tool's functionality to that which can coexist with other products.

Figure 5. Run Status window



DMW is driven by the various DAQ data bases, so as to cope with all possible DAQ system configurations. Given its slow real-time requirements (i.e. it should respond before the user becomes impa-

tient) such a module would greatly benefit from maintaining the DAQ status in a distributed data base management system.

3.3.3 Distributed Process Management

Process management is a major issue in a distributed, heterogeneous system such as the RD13 DAQ. While the possibility to create processes on remote machines is the minimal requirement, application flexibility needs more control on spawned processes such as the capability to delete created processes, to be (asynchronously) notified when they terminate and possibly to deliver UNIX signals. That is to say the major functions provided by an operating system such as UNIX.

Our initial process management was based on the UNIX remote shell (rsh) facility to create processes on remote machines. This was simple to implement but does not fulfil all of the requirements and leads to additional drawbacks in terms of performance and system resources (two additional processes are created by the rsh facility for each process created).

An application program interface (API) was therefore defined [TN128]:

1. To fulfil the process management requirements outlined above.
2. To interface to our "software description" data base [3] to identify programs by name, to retrieve static information (e.g. the machine where the process should run) and to store dynamic information (e.g. process id, state, etc.) if a truly distributed DBMS were available.
3. To support a user interface to monitor and control processes interactively.

A first prototype was implemented using the ISIS resource manager facility [3]. The batch oriented nature of the ISIS Resource Manager lead to an implementation suffering from a number of drawbacks, in particular performance and status information.

A second prototype was implemented based on the basic ISIS [15] toolkit. It consists of server processes, running on each of the DAQ machines and responsible for the process management on that machine. Commands and messages are exchanged between servers and client programs requesting their services. This implementation has acceptable performance and is being integrated in the RD13 DAQ.

This work has to be pursued to assess the functionality required, the integration problems and the issues related to scaling. While the solution adopted is satisfactory, a commercially supported product would be more suitable.

3.3.4 Run Control Open Issues

The system control area is a topic open to more study and development. A number of issues have not been investigated or need some re-assessment.

- More" intelligence" in the control system, to also look after incoherency that may occur as a result of user requested actions.

- We have to define a better signalling protocol between the users and the RCL system. The system's reaction delays are variable and it is sometimes difficult to know if the DAQ system is definitely stable in a given state, still trying to execute a command or engaged in an automatically triggered transition.
- We have also to assess the option of having the coherency management distributed, as it is today, or centralised.
- MACRO commands (a list of low level RCL command execute by operators or programs as one command) would also be a useful feature.
- On the user interface side, a simple end user interface to monitor the whole DAQ system would be helpful.
- A number of improvements could be envisaged to the current basic control software and to the control model. Such as more debugging facilities and improved system coherency management.

3.4 Data Bases

The management of information has continued to be a major issue for our research. We have focused our work on the evaluation of Object Oriented (OO) technology.

- The conversion of the existing RD13 DAQ data bases to use the ITASCA [47] Object Oriented Data Base Management System (OODBMS) was done. This includes both the schema design and the implementation of the data base access library [TN11]. A version of the DAQ using ITASCA for one of the data bases runs in our laboratory. This exercise allowed us to make progress in our understanding of how OODBMS can be used in our environment: in particular in the area of performance and software integration.
- An OODBMS has a rich set of features, more than required in our application field. We considered worthwhile to investigate the possibility of using OO techniques, such as those provided by a the OMW CASE tool [45], to provide persistence to application programs. Thus combining run-time performance (as given by our early QUID data base environment) with a powerful OO environment. Two new data bases, with associated utilities, were developed with the above objectives in mind: one to manage error messages and a second to maintain run bookkeeping information.
- OMW has also been used to implement a powerful browser for the hardware description data base [TN155]. The browser uses the access library to interface to the data base, it can therefore be used with either the QUID or the ITASCA implementation.

The technical details related to the points above are presented in section 5.

3.5 Central Data Recording

The central data recording (CDR) system, available from CN/PDP [6], was integrated into our DAQ in collaboration with the CN/PDP group.The implementation was driven by the following requirements:

- Support the foreseen test beam data rate of a few MB's per spill, leaving the option of a high performance (several MB's per second) CDR system to the future.
- Minimise the interferences to the existing systems, that is our DAQ and the CN/PDP CDR.

- Maintain the existing local recording in our DAQ and provide the possibility of switching between the two recording options by acting on the standard run control panel (e.g. by switching "recording devices").

Given the above requirements we decided to take the following steps.

1. A minimal coupling between DAQ and CDR; the DAQ writes files (named with the run number) on a local disk (which provides a local buffer of a few GB's). The CDR software accesses this disk and takes in input the files produced by the DAQ. Status and error messages produced by the CDR are routed to the RD13 error message facility via a simple program.
2. The use of ethernet as the transfer medium between the experimental area and the computer centre. The available bandwith is at the moment satisfactory and we did not see any need to immediately choose expensive solutions based on different protocols and media.

The CDR stages the run data files first on disk, on the computer centre side, and then copies them to tape streams (for example the same file may go to a 1GB IBM cassette, for analysis, and to a DLT tape for backup purposes). The CDR is configured locally (i.e. on the experimental area side) with labels of the tapes to be used and directives on the number of output streams.

The overall system performed on schedule and reliably. The local disk buffer helps to smooth both the peak bandwidth demand to a uniform value of a few 10s of KB per second and transient network problems when they arise.

3.6 Labelled Tapes

We have added a tape labelling component to the RD13 DAQ system [TN145]. The tape labeller allows the DAQ system to record data on labelled tapes according to the tape layout specified in [34]. The tape labeller accepts various types of tape (ExaByte, IBM cassette etc.), uses the DAQ's data bases to store and retrieve configuration information and is integrated with the run control system for synchronization purposes. The tape labeller is implemented as two processes (communicating via UNIX pipes) - a parent process that handles the connection with the run control system and a child process that controls the recording device. The slowness of some blocking tape device operations (such as rewinding) might cause time outs in the run control, hence the separation in two programs. A suite of programs are also provided for manipulating tapes offline.

3.7 Ancillary Tasks

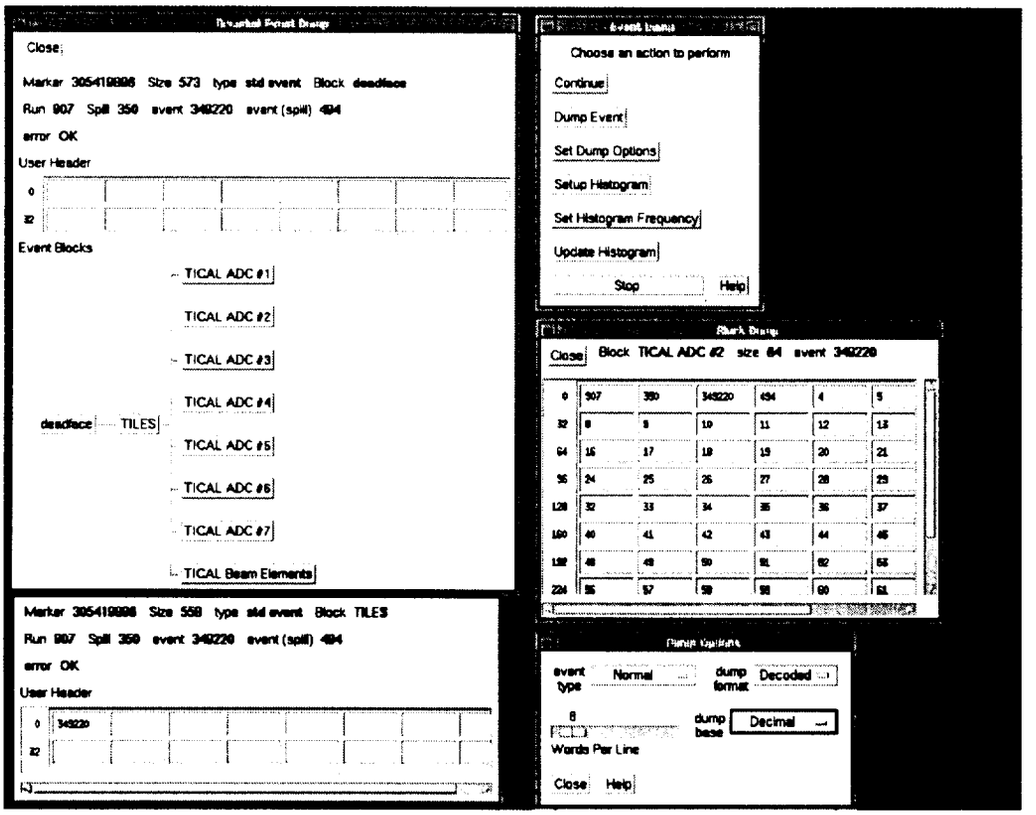
3.7.1 Event Dump

We have used the standard RD13 DAQ monitoring skeleton [TN46] to produce a graphical event dump [36] that can be used online to select and view events. It performs an event dump on request showing the structure and contents of a complete event. The user can define the event selection criteria which will cause events to be dumped.

To allow for sophisticated event selection, a simple language interpreter [35] has been integrated into the event dump that allows the user to write an expression which references the event structure and its contents.

Monitoring tasks are built on top of the skeleton in order to receive events from the acquisition system. The event dump is design to handle asynchronous "physics" event through the monitoring skeleton and also "graphical" events (e.g X protocol events) coming from the workstation X window server. The graphical user interface was developed using X-Designer [19] and a freely available tree widget.

Figure 6 Event dump showing event decomposition and data block contents

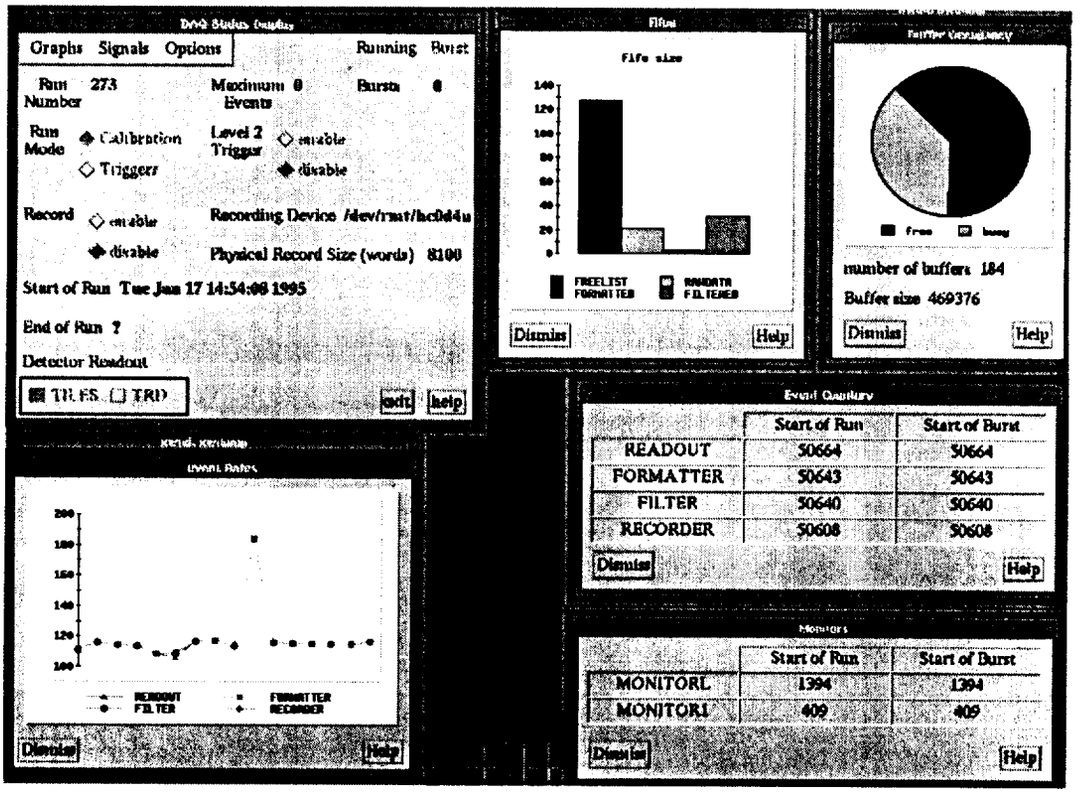


3.7.2 Upgrade of Status Display

We have developed a new version of the graphical status display (Figure 7) for the DAQ system which replaces the use of the DataViews widgets [20] with a new set of commercial widgets called XRT from the KLG group [7]. We have replaced the widget set for a number of reasons:

- The status display is built using X-Designer which now supports the use of the XRT widgets instead of the DataViews widgets. The X-Designer and DataViews integration is no longer directly supported by IST and would have become our responsibility.
- Having evaluated the XRT widget set we found them more powerful than the DataViews alternatives and with a programming interface that is more consistent with the standard Motif programming techniques.
- XRT offers a very powerful table widget that has also been used in the DAQ's message browser application.

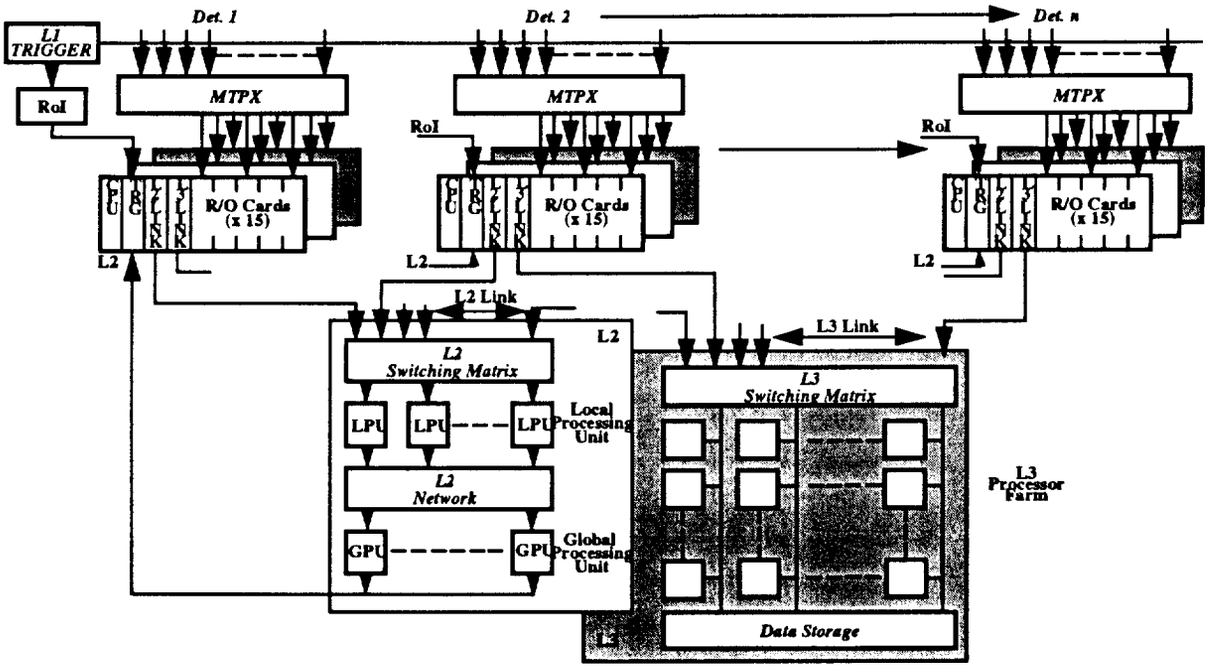
Figure 7 Graphical status display built using X-Designer and XRT widget set



4 Architecture Studies

We intend to keep our DAQ activity consistent with the study of a general LHC-like DAQ architecture. We follow as our working hypothesis the generic functional model of an LHC-like data acquisition system defined in [37] and sketched in Figure 8.

Figure 8 Global scheme of an LHC-like functional architecture.



To understand the requirements of such an architecture and to evaluate technological solutions, we think it is necessary to approach the problem from two complementary sides. Simulations to study models of the architecture and laboratory tests where small-scale versions of the architecture components, such as the event builder, are implemented as a combination of hardware and software solutions and tested.

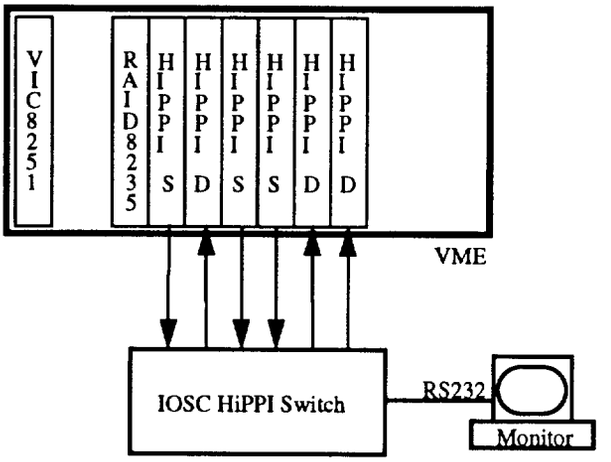
4.1 Event Building Prototype

Future experiments in high energy physics will need event building systems with a bandwidth of 1 to 10 GB/s and able to assemble event fragments from 100 to 1000 data sources at rates of 1 to 10 kHz. Bus based systems are inadequate, parallel event building based on high speed interconnects and switching elements will have to be envisaged.

A first prototype based on the HiPPI standard [58] was built as a small-scale version of a parallel event building system using commercially available technology. This prototype was used to study parallel event building, to integrate a particular set of hardware with generic software allowing for an evolution of the system in size as well as in different technologies, and to collect realistic parameters for further simulations of scaled up systems.

The whole prototype is housed in one VME crate with the exception of the switch [59] itself. VME-HiPPI interfaces based on the RIO module [60] act as either data sources (HiPPI/S module [61]) or data destinations (HiPPI/D module [62]). A RAID processor [63] provides the functionality of data flow processes and their control, as well as of monitoring the performance of the prototype.

Figure 9 Event Building Prototype Setup

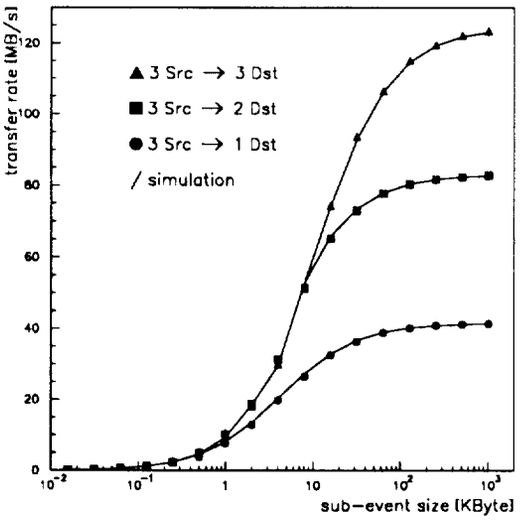


The software developed consists of two layers: the firmware and the data flow protocol. The lower layer [TN129], [TN143] is hardware specific and runs on the RIO module, sending and receiving event data by using the HiPPI protocol, one connection per event fragment. It communicates with the higher layer by using VME interrupts [TN130], [TN144]. The higher level layer is hardware independent and modular. It can be regarded as a stripped-down version of the RD13 DFP and functions as a mini-DAQ system. It contains data flow processes for the data sources and data destinations and provides the event building functionality of event assembly and destination assignment [TN111]. The event assembly component could be alternatively moved to the firmware running on the HiPPI/D module, while keeping the other modules on the RAID processor.

Simple data transfers from one source to one destination show, after optimization, a minimum latency of 49 μs. This overhead is mainly due to the firmware and the HiPPI protocol, the switch itself contributes less than 1 μs. The interrupt handling could be measured to be minimally 32 μs, so that the latency between data source process and data destination process is 81μs in total. The maximum frequency for sending events was measured to be 30.3 kHz, and for receiving events to be 23.8 kHz.

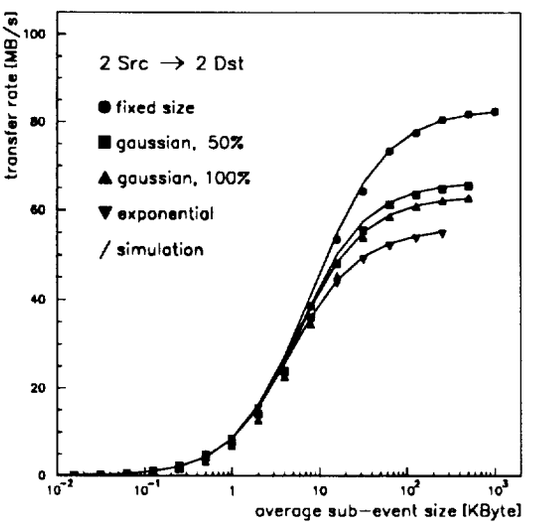
Several data sources and data destinations were combined to build a parallel event building system using a simple "push" algorithm for the destination assignment. The destinations were assigned in a round-robin manner and the flow control in the HiPPI protocol was used for synchronization, i.e. the transfer requests "camp on" the switch while the destination is busy. Figure 10 shows the maximum data throughput for different event fragment sizes, it scales with the number of data destinations for sizes above 10 kB. The lack of scalability at small fragment sizes is due to the sharing by the HiPPI channels of a single control process. Good agreement can further be seen between the measurements and the simulations carried out with the simulation program (see section 4.2) which uses parameters from the one-to-one measurements and two fitted parameters to represent the two data flow processes, the common control process and the interrupt handling (all parameters having values between 30 and 220 μs).

Figure 10 Prototype Performance



The influence of varying event size was measured as shown in Figure 11. Strong variations in the event size essentially disturb the parallelism and thus decrease the total throughput. Individual exponential event fragment size distributions can be regarded as the worst case which reduces the efficiency to about 76% compared to fixed event fragment sizes for a 2×2 setup. With only three source modules available the effect of fragment size correlations could not be measured.

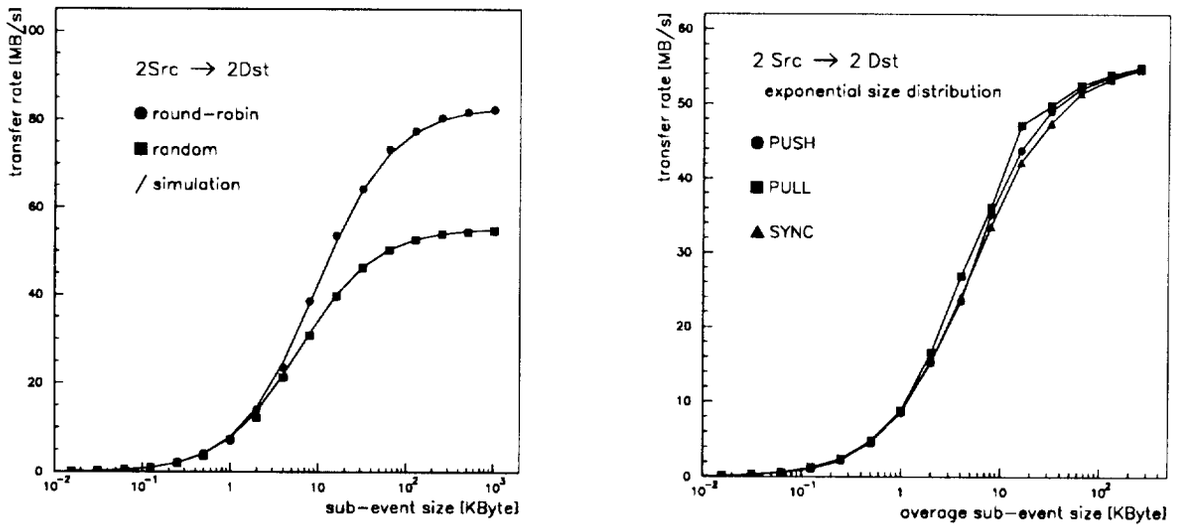
Figure 11 Influence of the Variation of Event Fragment Sizes



Different schemes for the destination assignment were tested. A random assignment of the destinations compares poorly with the round-robin scheme (34% less throughput for a 2×2 setup). Alternatively to the "push" scheme (*PUSH*) two "pull" schemes were tested which send the event fragments only after receiving a ready signal from the destination. The VME space was used for synchronization and the two "pull" schemes differ if they wait for the HiPPI/S module to have sent the previous event fragment (*SYNC*) or not, queuing the event fragments on the HiPPI/S module (*PULL*). No essential differences were found in the three schemes, as seen in Figure 12b). The existence of a

unique control process dealing with a different number of VME interrupts explains the observed discrepancy.

Figure 12 Different Control Schemes: a) Random Destination Assignment; b) PUSH,PULL,SYNC (see text)



The prototype has shown that parallel event building is possible using a commercially available technology. The HiPPI technology was chosen and integrated with generic software which can be extended to other technologies (such as FibreChannel or ATM) and integrated to other parts of a DAQ system for test beam purposes. Measurements with the prototype have displayed realistic parameters for the different overheads in hardware and software. With values of around 100 µs the simulations show a good agreement with the measurements.

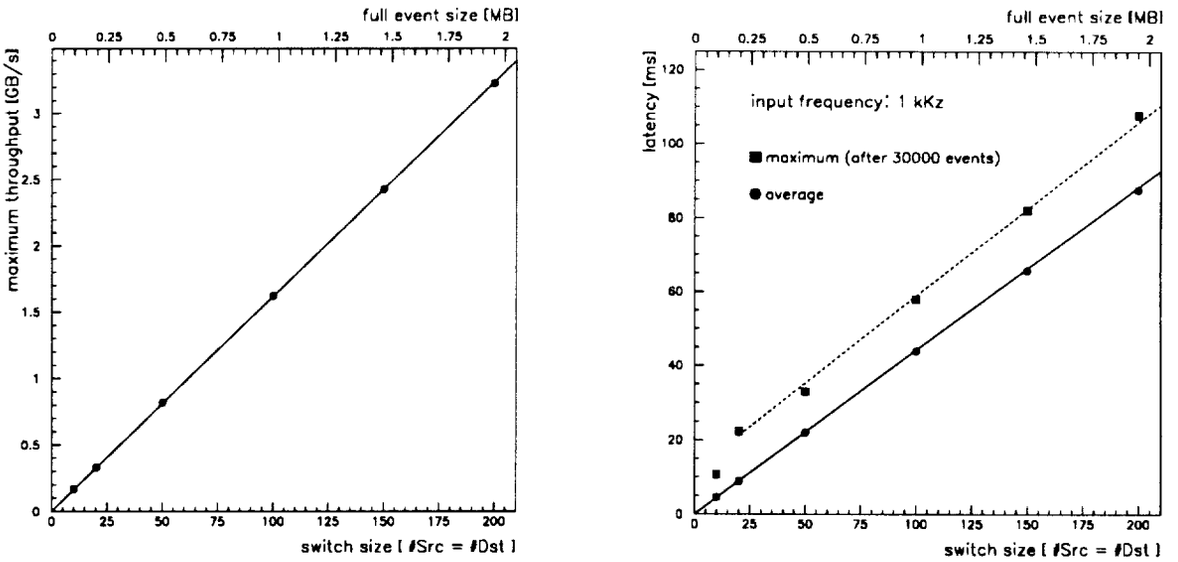
4.2 Modelling of Event Building Systems

Discrete-event simulations complement prototype building: prototype measurements help validating simulation models and these latter allow the study to be extended to scaled up setups.

All simulations for parallel event building systems were carried out with a program based on MODSIM II [64]. Originating from a simple version of this program [TN132] it was modified to be in accordance with the DAQ Simulation Library (DSL) [TN119] and can now be regarded as a subset of simplified DSL objects. It implements an event fragment generator, the data source and destination processes and a simple switch model which parametrises the transfer speed as a linear function of the event fragment size. This model can simulate different input event fragment size distributions, different control schemes and is fully configurable in number of data sources and destinations, as well as their parameters. It is appropriate for generic studies of parallel event building systems and was in particular used to simulate HiPPI and FibreChannel class 1 based systems.

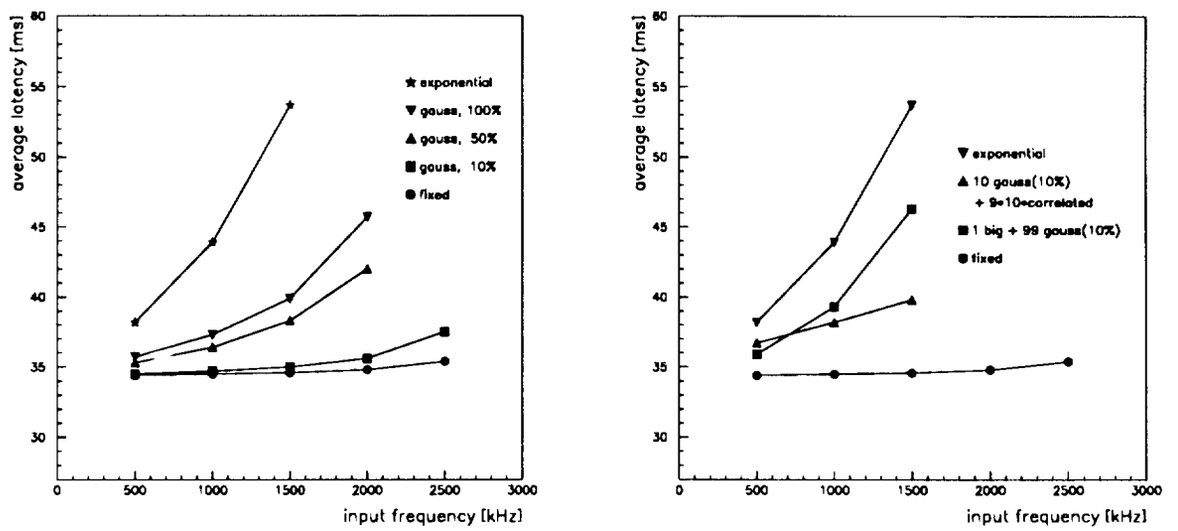
As described before (see section 4.1), the program shows good agreement with the measurements of the prototype. Generic studies based on the parameters from the prototype show that the maximum throughput of a switch scales with its size in number of sources and destinations, or equivalently the full event size. The average latency at 1 kHz shows a similar scaling behaviour.

Figure 13 Scaling of an Event Building System



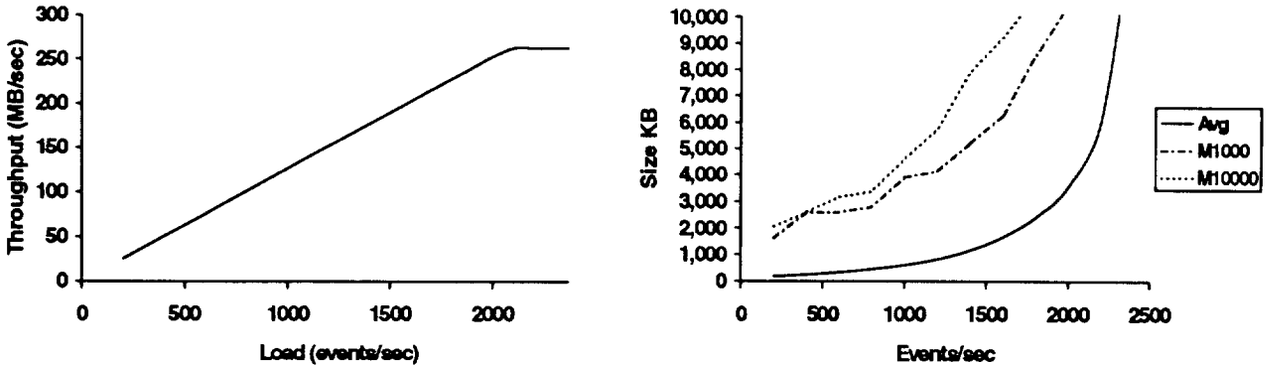
The influence of event fragment size variations could be simulated using different data models. Maximum throughput and average latency for different cases are shown in Figure 14a). Exponentially distributed event fragment sizes can be considered the worst case due to the long tails in the distribution. The correlation of event fragment sizes has a similar effect as can be seen in Figure 14b).

Figure 14 Influence of Event Fragment Size: a) Variations; b) Correlations



An alternative approach, smoothing out the event fragment size distribution, is to cut down the event building system into two stages [TN132]. A simulation comparing a single 256×256 switch with a two-stage system with 16 switches of 16×16 ports on each stage, and with an exponential event fragment size distribution reveals that the efficiency of the switch usage is about 70% higher in the two-stage case.

Figure 15 Two-Stage Event Building System: a) Throughput for First Stage; b) Buffer Size for Second Stage



The influence of different control schemes like in the prototype (i.e. *PUSH*, *PULL* and *SYNC*) under realistic parameters and with realistic event fragment size distributions from a detector simulation [65] will be investigated and studies are under way. Another field of investigation is the influence of different control schemes at the level of the individual event fragments. These algorithms known as "traffic shaping" for packet-oriented networks [66], could be implemented in this domain of connection-oriented networks in the following way: an event fragment is held if its destination is momentarily busy while the fragment of another event is sent instead.

4.3 Modelling of the ATLAS Second Level Trigger

Members of RD13, in collaboration with RD11, RD24, RD31 have developed a package (SIMDAQ) [40], written in MODSIM, to evaluate the performance of the proposed ATLAS second level trigger (T2) architecture [39].

The basic structure of the SIMDAQ package is a representation of the generic building blocks of the proposed architecture, that is the read-out buffers, switches, T2 supervisor and local and global processors. Detectors are mapped to the read-out buffers by configuration files, and input data, originated from the GEANT/PYTHIA suite, is mapped to the buffers in the detector's coordinate system. The generic model may be augmented with hardware specific models. Thus evaluations have been made with parameterized models of ATM and SCI for the switches. In the future other candidate technologies can be mapped to this architecture. Figure 16 shows results for a simulation run with ATM622 and SCI showing buffer occupancy plots for the read-out buffers. The latency of 1.8ms is dominated by the chosen processing times for feature extraction in the modelled detectors (300 microseconds for

the e-m calorimeter, 300 microseconds for the hadron calorimeter and 700 microseconds for the TRT detector), and the global processing time of 500 microseconds.

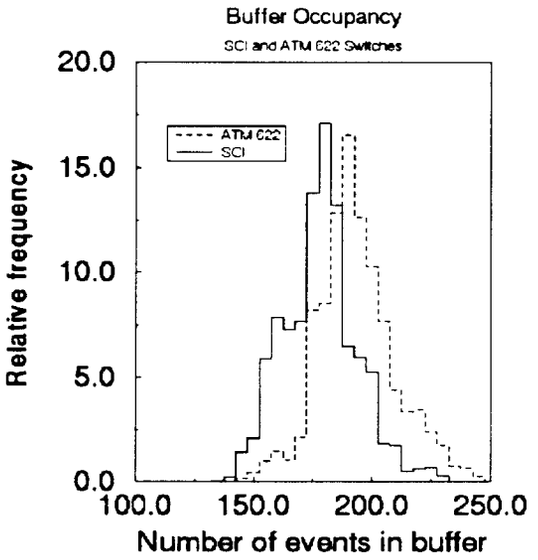


Figure 16 Read-out buffer occupancy in ATM and SCI models

The work so far has confirmed the expected performance with a simple push model and a round robin allocation by the supervisor of the local and global processors. The domination of the processing time has put strong emphasis on developing more detailed data dependent parameterized models of the processing, together with the generation of larger physics data sets including data for calorimeters and TRT with the addition of the muon and other tracking detectors.

Work is continuing in several other areas, such as alternative T2supervisor models and detector dependent models of the data arrival in the read-out buffers.

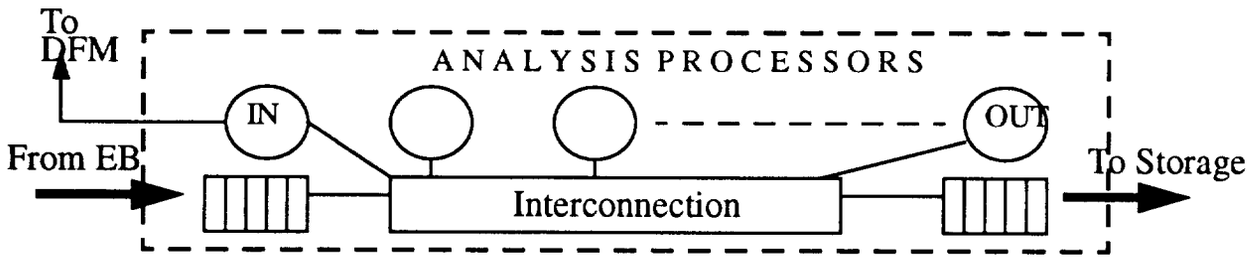
4.4 Level-3 like System

A generic view of a level-3 system [39] is that of a set of independent sub-farms, each connected to an output of the event builder and consisting of a number of processor units (CPUs). Each sub-farm CPU is assigned an event for full processing. A functional sketch of the sub-farm is shown in Figure 17, where the basic *functional* elements of the sub-farm are highlighted:

- Input segment: which receives full events from the event builder (EB), buffers them in memory and provides feedback information to the EB data flow manager (DFM). It consists of a "processor" with a "buffer" and links to both the EB data channel and the DFM.
- LVL3 processing elements: each processing element runs an analysis task which receives full events from the input segment. Selected events are passed to the output segment.
- Output segment: it receives selected events from analysis tasks and sends them to the permanent storage system. It consists of a "processor" with "buffer" memory and links to the permanent storage system.

In the above description terms such as sending and receiving, processor and buffer indicate functions, without reference to a particular realisation.

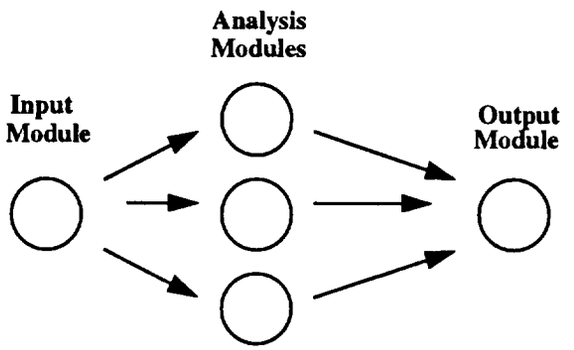
Figure 17 Level-3 sub-farm, functional view.



The data flow view of the sub-farm is sketched in Figure 18, arrows indicating the logical flow of data (events) from the input to analysis and eventually to the output modules. While the two above views of a sub-farm are generic, choices have to be made in terms of architecture (how processors, memories and interconnections are organised) and size (e.g. how many CPUs are needed per sub-farm) so as to fulfil both the performance requirements and the financial constraints.

These problems can be studied by a combination of modelling/simulation techniques and small scale prototypes. For the latter, we remark that the work done by RD13 in the area of data flow for DAQ systems can be re-applied to the problem of managing and distributing data in a level-3 sub-farm. Figure 18 can be re-read as a DAQ-Unit [3] with multiple, concurrent data handling modules.

Figure 18 Level-3 sub-farm, data flow view.



On the basis of the above considerations, we have initiated the study of the organisation of a stand alone (i.e. without reference to its interface to the event builder) sub-farm system from the point of view of its performance. Analytic modelling has been used to study some simple (bus based) architecture. We have also initiated to set up a framework prototype to run real level-3 like ATLAS programs on alternative multi-CPU architectures.

4.4.1 Analytic Modelling

Performance analysis by analytic techniques is an inexpensive complement to time consuming simulation methods. While they are obviously limited by the very simplifying assumption needed to derive the solutions, they may constitute a useful tool to prune the design space, before other methods are applied, and to provide useful insight into the problem being studied. We have used such techniques in two ways: the analysis of a generic sub-farm and the study of the performance of a shared bus based multi-processor system.

Buffer and sub-farm sizing

The initial investigation of the requirements in terms of input buffering and number of CPUs, as a function of the input event rate, can be done by means of simple queuing models [41] under the following simplifying assumptions. The event arrival and processing rates are exponential, event data transfer and output treatment times are negligible with respect to the event processing times. If the input buffer is assumed of infinite size, the simple exponential multi-server queue (M/M/k) can be used to derive average queue sizes, mean response times, etc. In the case of a finite input buffer a corresponding queue (M/M/k/B) can be used to derive e.g. the probability for the input buffers to become full.

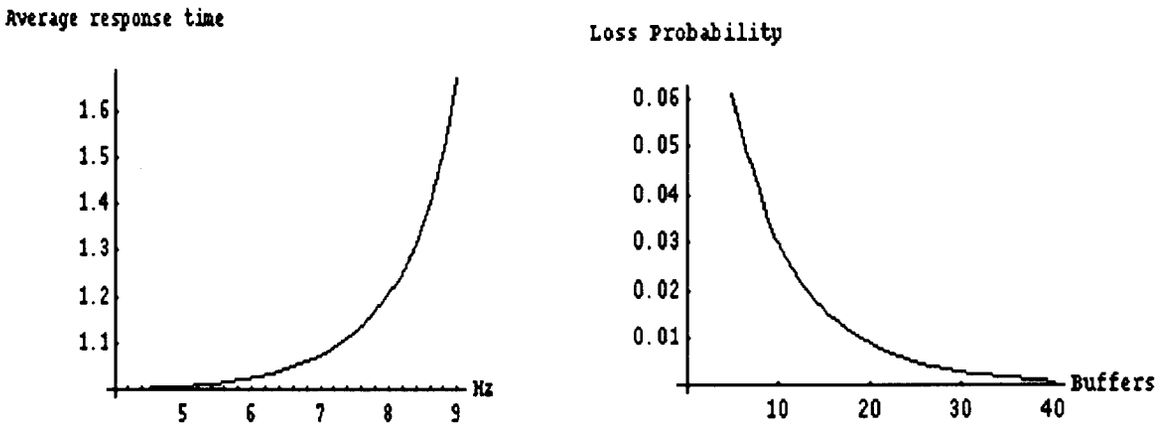


Figure 19 (a) M/M/k average response time (sec.) (b) M/M/k/B loss probability

Figure 19 (a) shows a graph of the M/M/k average response time (in seconds) for the case of a sub-farm with 10 processors, average processing rate of 1 Hz and an event arrival rate varying between 5 and 9 Hz. Figure 19 (b) shows the M/M/k/B loss probability for a sub-farm with an arrival rate of 9 Hz, a processing rate of 1 Hz and 10 CPUs as a function of the number of input buffers.

Shared bus systems

An analytical calculation [42] has been performed for the simplest level-3 sub-farm architecture - a multiprocessor system, where each processor has a CPU and local memory. The processors are tied to a common (shared) memory via a single bus. Each processor runs the same application, i.e. the same analysis program. Performance evaluation must be done to determine the most efficient, and therefore

cost-effective, use of the shared resource (memory). A performance index of such a system is the average number of active processors, i.e. those which are not stalled because of contention for the shared memory. The ratio of the number of shared memory accesses to local activity (processing and access to local memory) is the sensitive model parameter. For e.g. a 10-processor system, results show 20% inefficiency for an application performing 10% of its accesses to shared memory. The inefficiency grows to 40% at 20% shared-memory accesses.

4.4.2 Level-3 Program Parametrisation

When applying any kind of modelling techniques to a problem such as that of the level-3 sub-farm, the proper parametrisation of the analysis program itself is obviously a key element. Under the assumption that one processor runs one level-3 program on one event at a time, parametrization would require:

- average number of instructions executed per event: to obtain a coarse idea of the execution time.
- memory requirements: to size the sub-farm memory system (in particular in the case of a shared memory system).
- cache behavior: to parameterize the use of multi-level memory systems.
- memory interconnection utilization: requirements on the processor-memory interconnection system (e.g. a bus) are defined by the load (instructions and data) of the program execution.

The values of the above parameters, in the form of averages or statistical distributions, ought to be measured on real programs. To this end we have investigated how a framework, where such measurements can be carried out, can be setup by using both real programs and proper tools.

Our initial work [TN156] has been carried out based on an ATLAS reconstruction program, with a well defined set of input events in raw data form. The WARTS [43] toolkit has been used to instrument the program executable to obtain both dynamic instruction counts and memory traces. These latter are then fed into WARTS cache simulators which measure both cache performance and bus load.

4.4.3 Prototyping

Starting from the core of the RD13 DFP software [TN15], we have derived a skeleton system to distribute events in a bus based system such as that consisting of our RAID 8235 processors in VME. The skeleton is layered to isolate the system dependent part (e.g. process synchronisation), so that, in principle, it could be adapted to other, non bus-based, architectures.To use this software in a realistic way, we will employ the same ATLAS program used for the work described in section 4.4.2. Work to port such a program first to our front-end VME processors and then to integrate it with the DFP-like skeleton is in progress.

5 Object Oriented Software Technology

Two commercial products supporting object oriented software had been selected in the previous phase of RD13 [3]: the ITASCA [47] object oriented data base management system (OODBMS) and

the Object Management Workbench (OMW) [45] CASE tool. Both products were heavily used in RD13 during the last year.

5.1 ITASCA

5.1.1 Applications

The activities around ITASCA have followed three major themes: data modelling (to understand how best to exploit ITASCA's object oriented capabilities), data base conversion and their integration in the whole DAQ system (to evaluate the use of ITASCA in an online environment and compare it with the solution provided by a tool such as QUID) and the development of ITASCA based application outside the current data base framework (to program directly with ITASCA without going through a C-like, non object oriented, data access library interface).

A number of internal notes, [TN127], [TN134], [TN135], [TN 149] and [TN151], document the exploratory work done in the area of ITASCA.

Data Modelling

The conversion of the existing data bases provided the ground to exercise the data modelling capabilities of ITASCA.

A simple data base such as the one describing the run control finite state machines was translated directly from the QUID entity-relationship (E-R) model into ITASCA's object oriented (OO) model. The DAQ software components data model was instead re-designed from scratch both to take advantage of the OO model and to add additional features (such as the introduction in the data base of more DAQ components and a better support of the process manager functions). A detailed account of this work is documented in [TN135]. The hardware description data model was also re-designed to be conceptually much simpler and easier to use.

Integration of ITASCA with the DAQ system

The integration of the data bases and their access libraries, implemented with the ITASCA client application program interface (API), proved to be a challenging task. A number of integration problems arose from the synchronous message based client/server implementation in the ITASCA client clashing with the asynchronous, programming model suggested by ISIS. This implied non trivial modifications to the client application, in particular the run control component, to be protected for example from race conditions in the access to the data base. The situation was further complicated by the lack of proper debugging tools for distributed, event driven applications.

We also had to retrofit into ITASCA the programming model previously implemented with QUID, this showed to be inefficient when dealing with a distributed data base (as opposed to a memory resident one). This resulted in poor performance, which could be improved to an acceptable level by using the client-side caching mechanism of ITASCA.

Although one of the data bases, the one holding run control parameters and parametrising the read-out, was successfully integrated an the resulting DAQ system run in the laboratory, the integration problems mentioned above delayed our programme of work.

ITASCA Applications

The Message Loader and Browser [TN134] is a new application that uses ITASCA as a means of storing a log of all the error messages generated by the DAQ. This application required the integration of the ITASCA client library with the ISIS based run control system (though a data access library was not used as in the applications described above). A separate browser application was developed in order to view the recorded messages. The applications were separated for the following reasons:

- Performance. It is possible to start the application on different workstations (e.g. run the Message Loader on the same machine as the ITASCA server to reduce network traffic) or with different priorities or start only the Message Loader if online browsing is not required.
- Reliability. The Message Loader application is now as simple as possible. The XDesigner tool with the XRT table widget was used to generate the graphic user interface for the Message Browser and installed Run Control reception filters in the Message Loader. There is also a specialized separate program to install the data base scheme in ITASCA.

5.1.2 Conclusions

Substantial effort went into the area of OODBMS. While we were not able to perform fully the research programme, we have acquired concrete and valuable experience in the area of OODBMS and in their application to typical DAQ problems.

The selected tool, ITASCA, maintained its promises of a sophisticated, state of the art OODBMS. Client-side caching proved to be an important feature to obtain acceptable performances also in the part of the real-time side of the system. Its inexpensive availability on our EP/LX and LynxOS based platforms was an additional advantage.

The integration problems we experienced were not necessarily due to the tool itself, rather they were the result of trying to combine components (the data access library, the ISIS based application program and the client/server oriented data base) with at times contradictory and incompatible requirements and implementations. These problems do arise when integrating commercial products which may look functionally coherent and complementary but are often incompatible implementation-wise.

ITASCA, like most other OODBMS, is a full data management system, it includes the concept of transactions and locking for concurrent access to shared data. These features are not always necessary, in some cases data are shared in read-only mode (e.g. configurations and parameters during a DAQ run) thus making redundant most of the run-time features of the system. A careful study of requirements for data sharing should be undertaken, and lighter, more efficient solutions might be envisaged for data shared in read-only mode (e.g. some memory resident copy is loaded from the full data base at initialization).

5.2 Object Management Workbench (OMW).

OMW supports the Object Oriented Information Engineering (OOIE) method [54] and provides a number of tools to fully support the software life cycle from analysis through to code generation and testing. It integrates model definition (via object and event diagram editors), model behaviour constraints (via a rules editor), user interface building, model testing and simulation (scenario manager) and document generation (via a report browser).

The OMW tool set is built on top of the Kappa programming environment [55]. The Kappa system is an ANSI C-based visual environment for developing and delivering distributed applications in open systems environments. Kappa applications can also be distributed, using the Kappa CommManager, between UNIX workstations and PCs running Windows. The CommManager provides transparent communications among distributed objects by running over TCP/IP networks and complies to the CORBA protocols defined by the Object Management group (OMG)[56].

The OMW/Kappa development tools run on Sun (SunOS and Solaris) and HP workstations. The developed applications can be ported to IBM compatible PCs running Microsoft Windows.

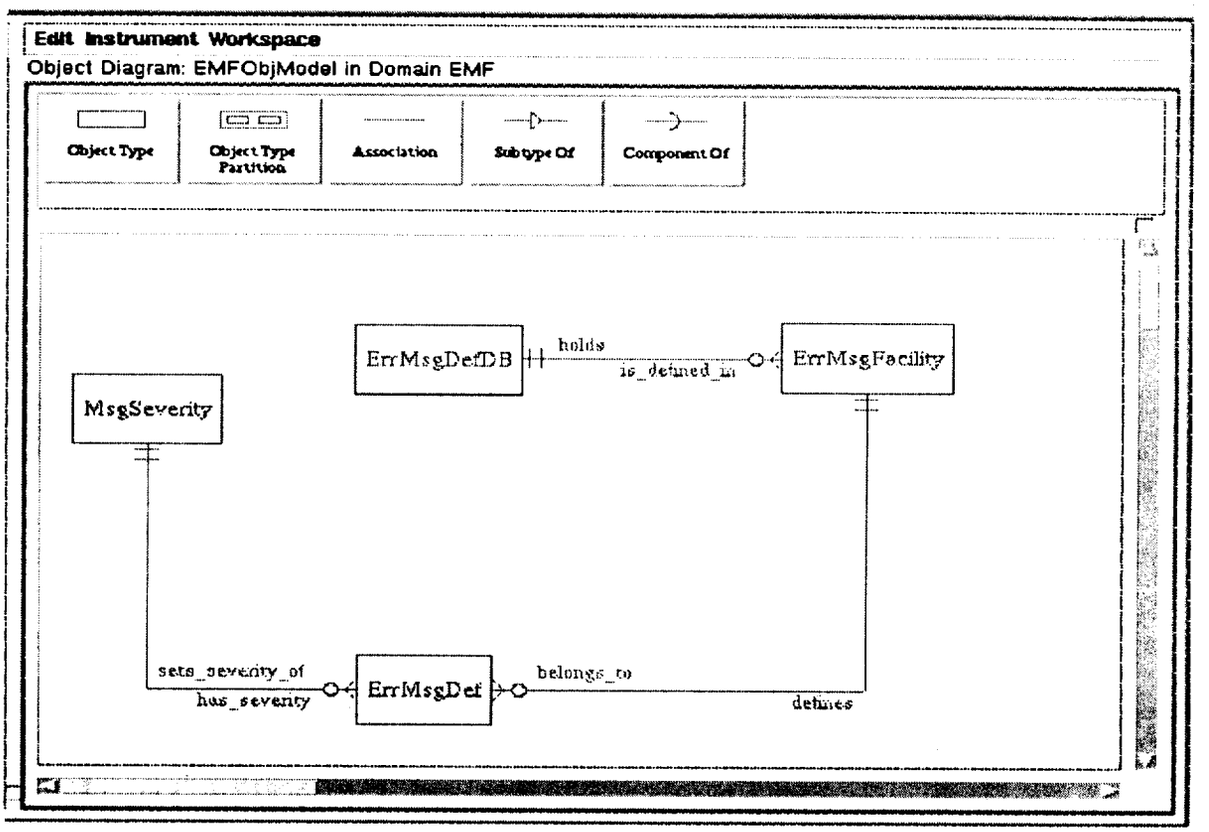
5.2.1 OMW Applications

Error Message Definition and Translation

We have used OMW to develop a replacement for our existing tools for producing and maintaining unique error codes within the DAQ software. Previously we have used the GENMESS facility (a porting of the VMS MESSAGE facility to UNIX) [27] to define unique error messages. This had several limitations including no automatic means of avoiding collisions between error codes defined in separate files.

We have replaced this system with a data base of error messages implemented using the OMW/ Kappa internal object persistence mechanism. Figure 20 shows the object diagram for the error message data base. Two application programs work on the data base - the EmfEditor and EmfTrans.

Figure 20 OMW Object Diagram for Emf error message data base



The Emf Editor program allows the user to interactively modify the definitions of error messages used by the RD13 DAQ system. The user can modify existing error message definitions, define new messages or delete obsolete ones. A C include file containing all the definitions can be generated. The definitions are loaded and stored to disk files and are also used by the EmfTrans application.

EmfTrans is part of the EMF error message facility. It is a server process that performs the translation from error code to textual error message. Application programs exchange messages with EmfTrans just before they report error messages, or when they simply want to retrieve the text associated with error codes. EmfTrans is implemented as an OMW/Kappa application that loads the data base of error message definitions and connects to the distributed run control system based on the ISIS communication package. When it receives a translation request from DAQ processes it looks up the given error code in the data base and returns the associated textual message.

Online Volume Bookkeeping

The Online Volume Bookkeeping (OVBK) system is designed to provide an automatic log of the data recording by the DAQ system. The OVBK retrieves information from different DAQ modules and data bases to complete the log. The data describing the run configuration, run date, archive file information and run quality are maintained in a data base. The OVBK system also provides a graphical user interface to interactively access the data base and to generate reports. This application required the integration with run control system, the run parameters data base and the error message facility. It

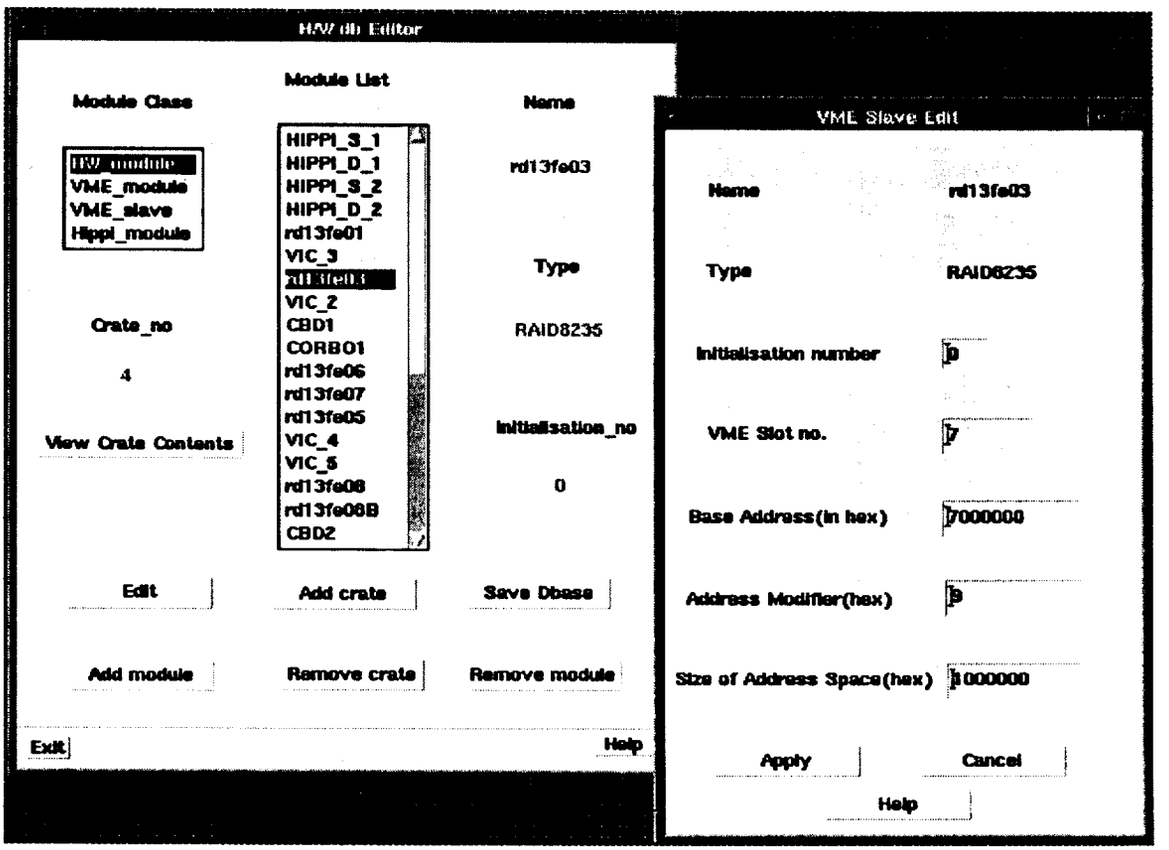
is implemented as a process that runs on a workstation while the DAQ is taking data and writes the details of the current run to a data base on disk using the OMW persistent objects facilities. The graphical user interface is run as a separate process off-line.

Hardware Data Base Editor

The hardware data base editor (hwdbEditor) [TN155] program allows the user to edit the contents of the hardware data base used by the RD13 DAQ system. The user can modify the existing configuration, define new hardware modules or delete obsolete ones. The aim of this application is to replace the standard QUID editor which requires a knowledge of how the data base was implemented in terms of the entity-relationship (E-R) schema.

The hwdbEditor application uses the Object Diagrammer to make an OO equivalent of the QUID E-R model. The actual contents of the hardware data base are loaded at run-time from QUID, using the data access library, and instances of corresponding the object types are created. The user can then manipulate these instances via a graphical user interface(Figure 21). Finally the modified contents are written back to QUID onto disk via the access library.

Figure 21 OMW based hardware data base editor



OMW/ITASCA interface

As an evaluation, we have developed a prototype application that provides a link between the OMW CASE tool and the ITASCA OO data base. The motivation for this prototype was to see if an alternative technique for object persistency could be used with OMW and also to provide a graphical schema editor for ITASCA. The prototype is implemented as an OMW application that uses the Kappa library to access meta-data about object diagrams and ITASCA's C client library to access corresponding definitions in data base. The prototype can define classes with attributes and relationships including inheritance and cardinality constraint then store OMW object instances in ITASCA and reload the instances from ITASCA to OMW. The prototype highlighted a number of differences between the respective object models of the Kappa environment and the ITASCA data base. A limitation with such a technique is that the OMW object methods could not be easily stored in the ITASCA server because they are not written in LISP. But there is one important advantage of using ITASCA over OMW/Kappa internal object persistence - the objects from ITASCA can be saved and restored on demand. With Kappa's object persistence all the objects of a domain are either stored or retrieved in block.

We have shown that it is possible to provide such a link between the CASE tool and the data base and that such a technique could possibly be used as a means of CASE tool independence or migration (e.g. store applications developed with OMW in ITASCA then read it back out by another CASE tool).

5.2.2 Assessment of OMW

We have found the OOIE method clear yet sophisticated with sufficient support for static object definition. There are sufficient constructs in the event diagrams to represent synchronization and parallelism. Through our prototype interface between OMW and ITASCA, we found that the OOIE object model as supported by OMW and Kappa is very rich and that many features could not be implemented in ITASCA (e.g. user-definable Object Identifiers, monitors and slot formulae).

OMW does not fully implement the method, for example finite-state diagrams as an alternative to event diagrams to represent object behaviour are not supported. Some other features of the method are only partially supported by the CASE tool, for example event diagrams are not re-entrant (hence they cannot be used recursively).

The incremental development cycle support by OMW is very practical and one of the best features of the tool. One of the most important advantages of using OMW is the clarity of the diagrams produced (i.e event and object diagrams). The impression of how an application is structured can be gained by a simple inspection of the diagrams. The diagrams are guaranteed to be up to date and complete since the run-time code is generated from them.

Also, the principle of object domains allows the developer to modularise the application into groups of closely related object classes. OMW is a very open tool as shown by the number of third party software packages that we have been able to integrate with our applications.

The built-in simulator allows applications to be tested before generating a run-time (i.e. the resulting application executed outside of the development environment), so that the most obvious bugs can be

removed without leaving the tool. In general we have found that those bugs which persist past this stage of testing are usually related to integration with third party software (e.g. ISIS or other DAQ components) and effects of speed and space differences between interpreted and compiled code (the simulator interprets the application code).

The underlying Kappa programming environment is very rich and sophisticated. It implements a proprietary language (ProTalk) which can be mixed with C in methods. While C is essential to e.g. access operating system features, ProTalk is very powerful to deal with the object model and to perform rule based programming.

Kappa integrates a graphical user interface builder which offers similar facilities to X-Designer but has one important advantage - the possibility to link application objects to graphical objects without programming. The relation between the application and graphical objects is maintained so that, for example, if a new instance of an object class is created it can be made to appear automatically in a list widget. To provide a similar behaviour with X-Designer the developer would be required to write many callback routines to move the values from the screen and back to the application objects again. However, we have found that we cannot integrate third party widgets so easily as one can with X-Designer.

We have shown that it is possible to integrate OMW with third party archive systems such as CVS [49]. The need for a run-time library restricts the platforms on which we can run our generated applications. This means that we cannot use OMW to develop applications which must run on our front-end processors (e.g. RAID boards running EP/LX or LynxOS). The availability of VME boards running workstation-like operating systems (such as HP-UX or Solaris) may provide a solution to this problem.

The internal object persistence is very useful and has been used as the basis of the Emf and online bookkeeping data bases. The object diagram editor plays the role of a data base schema editor. The system also has the advantage of providing limited schema evolution - that is to say that objects can be saved to disk, the schema changed and the data base re-read into memory. The facilities of the object persistence cannot be compared to a real data base (e.g. there is no notion of transaction support) but it is more akin to an OO version of QUID.

We have performed limited tests with the CommManager package for distributed applications but do not currently use it in any of our applications. It appears to work satisfactorily on a network of SUN workstations but we have yet to test it in a heterogeneous (HP and SUN) environment. The programming overhead of distributing an application over several processes is minimal. We are particularly interested in combining the CommManager with the object persistence and object monitors in order to provide a distributed, reactive data store.

There are a number of problems and bugs in the CASE tool that have slowed down the development of the applications. In general we have found that the interpreter is the source of many problems. The tool requires a lot of resources in terms of memory and swap space which restricts its use to the more powerful workstation configurations and means the startup-time for the tool is quite long.

6 Commercial Software Products

One of the main objectives of RD13 is the study of the integration of commercial software products. We have incorporated many commercial products in the development of the DAQ system. Table 2 lists all of the products currently in use or under evaluation by RD13.

Table 2 Commercial products

product	status
ISIS: A toolkit for distributed and fault-tolerant programming	in use
Artifex: A CASE environment for the production of event-driven systems	evaluated
Quid: A simple single-user data base system based on Entity-Relationship approach for use with C programs	in use
ITASCA: Object-oriented data base management system	under evaluation
GemStone: Object-oriented data base management system	evaluated
StP (Software Through Pictures): A set of CASE tools supporting various form of SA/SD methodologies and limited code production	in use
FrameMaker: A WYSIWYG text processor with hypertext capabilities	in use
Purify: A package for the detection of run-time memory leaks inside programs	in use
Motif: A graphical user interface toolkit based on the X Window System	in use
X-Designer: A graphical user interface builder that allows the user to construct MOTIF interfaces to applications	in use
DataViews Widgets: A collection of graph widgets to display the data incorporated with X-Designer	evaluated
XRT Widgets: A collection of graph widgets to display the data incorporated with X-Designer	in use
LabVIEW (Laboratory Virtual Instrument Workbench): Software package to simplify computation, process control, and test and measurement applications	in use
MODSIM II: Simulation language with support for object-oriented programming and discrete event simulation	in use
Logiscope: A software analysis tool to understand the structure of a program and assess the quality using metrics	in use
ObjectCentre: Programming environment for C++ (editor, debugger, compiler, interpreter and code structure info.)	in use
Object Management Workbench: Object-oriented methodology and CASE tool	in use
XRunner: A tool for testing graphical user interfaces	evaluated
T: A tool for generating test cases	evaluated
Process Weaver: Process management tool	evaluated

PART II

RD13 Project Assessment

7 Assessment of the RD13 DAQ

RD13 has produced a line of prototype DAQ systems, of increasing complexity and functionality to match the requirements of the detector setups, which have been used in test beam runs.

The prototype to be used in the test beam is only one of the RD13 objectives, yet it provides the ground where the technical issues explored and the results achieved by RD13 may be assessed. To this end we consider the DAQ system according to a number of "quality" criteria.

Functionality

The capability to scale with requirements is the key characteristic of the system. This function is designed into the system at two levels: at the level of a single component (detector, full event processing unit, etc.) where processing power may scale according to requirements and at the global system level where the number of data sources may scale according to the setup. Scalability is supported by a number of design concepts and system features. The data flow and control components were designed with scalability as the primary requirement, the data base driven configurability makes changes in configuration transparent to the DAQ software. Scalable commercial software tools, such as ISIS [15] and the real-time UNIX system[22], are essential and the underlying data transfer hardware is inherently scalable. This latter determines the overall performance of the system, yet the design allows to scale in performance by changing the data transport hardware (for example switching from the VIC to the FDL system).

A second function which is related to scaling is the possibility of splitting the system into sub-systems of different size, with equal functionality and capable of running concurrently (partitioning). RD13 did not fully explore such an issue, although the fact that data bases describe the system externally and the data transport hardware and the distributed programming environment (ISIS) are partitionable makes partitioning possible. Indeed some very crude form of partitioning was used in September 1994 when both a combined run and a single detector run could take place concurrently.

Technology Independence

Understanding the issues related to the design and production of a system which may stand changes in hardware (e.g. processors, data transfer links) and software (e.g. operating systems) is compulsory for systems such as an LHC-like DAQ. Modularity and adherence to hardware and software standards are pre-requisites for success, yet the system has also to be designed with technology independence in mind. We consider that the RD13 DAQ system fulfils the objectives defined in [1]; software standards have been used where available (e.g. UNIX, Motif), the software has been produced with

(UNIX-like) platform independence in mind, hardware peculiarities have been hidden by layered libraries.

Flexibility

The adaptability to changing requirements is a another key point of the system.

The system can be configured to run in a single detector (with or without an event builder stage) or in multi-detector mode; for test purposes a basic skeleton is provided to users to fit their specific read-out code. The system can be easily adapted to setups of different size and requirements.

The clean separation of the detector read-out from the general data acquisition logic has allowed running with detectors using very different front-end electronics. New components, in particular detector DAQ systems, have been integrated with little effort.

Data base modification is the main activity required to re-configure the system. The data base system currently in use, QUID, automatically produces editors, which use the schema as the basis of their user interface. This makes the use of the editor both complex (the schema is designed with data modelling and efficient navigation in mind) and requires detailed knowledge on the data base internals. Powerful, user oriented graphical editors are needed to support flexibility. We have experimented in this direction with the OMW [45] object oriented programming environment (see section 5.2.3).

Reliability

Despite the fact that we have produced prototypes from laboratory R&D work, therefore not engineered with reliability as a primary requirement, the detectors using our systems have experienced very few problems on the DAQ side. Hardware failures (e.g. disk crash) have been the main source of inefficiency.

Performance

The performance of the RD13 DAQ can be discussed in terms of basic overheads, to determine the cost of the system features, and of its capability when taking data in a test beam run, to verify its real life behaviour.

The data flow protocol (DFP) basic performance was reported in [2]. We just remind that in a configuration with two DAQ modules (a dummy[1] read-out, and a dummy recorder) the system can run, in continuous mode, in excess of 3000 events/second, with a measured overhead of less than 300 microseconds for the execution of the DFP in the read-out module.

We have performed laboratory measurements with the current software event builder (EB) in a configuration including one local DAQ producing dummy events (i.e. events consisting of only the main header, 52 bytes) and a central DAQ with 3 DAQ modules (the event builder, an event distributor for sampling, and a dummy recorder). That is the basic configuration used when taking data in the test

1. The term dummy in this paragraph refers to a DAQ module which is functionally complete but performs the minimum possible data transfer. For example the read-out deals with only the event format but no real data, the recorder skips the actual writing to data storage.

beam in multi-detector mode. It is set such that all the (detector independent) overhead is taken into account (both on the local detector DAQ and on the EB side). Figure 22 (a) shows the number of events per spill (2.4 seconds long) treated by this DAQ configuration as a function of the available triggers per spill[1]; a saturation value of about 6800 events/spill is achieved.

Figure 22 (b) is instead related to the data transfer capability of the system. The DAQ is configured as for Figure 22 (a) but now the local detector DAQ produces events of different length at a fixed trigger rate. Events are read out with either the VIC based EB (using program controlled transfers) or the FDL based EB.

Figure 22 RD13 DAQ rates with 1 local detector. (a) triggers (b) data transfer

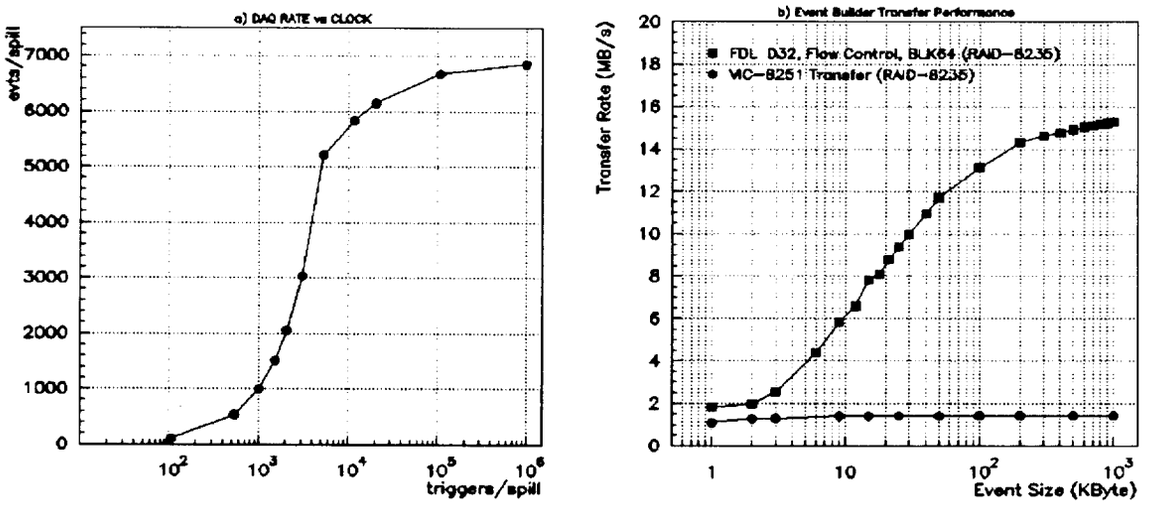


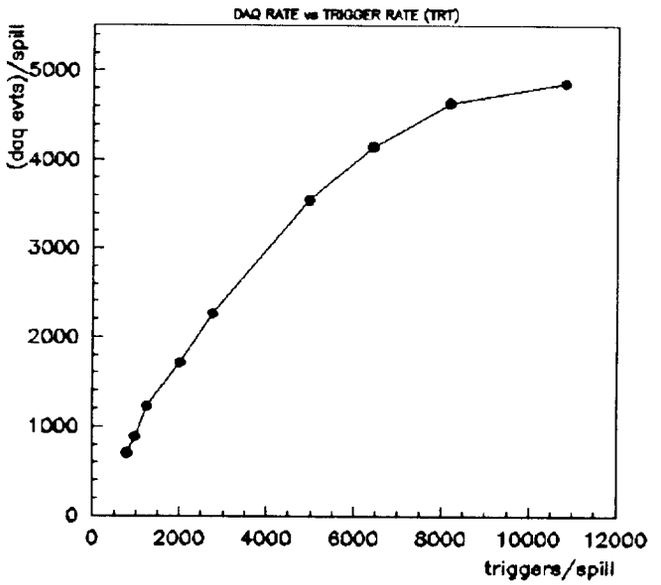
Figure 23 refers to measurements with real beam taken by RD13 in June 1994 during the TRT run (see [3] for a description of the setup[2]). The DAQ was configured in single detector mode (i.e. without the EB stage) with two DAQ modules: a "read-out", reading two TRT sectors (136 data words per sector), and a recorder. The graph shows the number of events taken per spill as a function of the available trigger rate (varied by acting on the beam collimators). The system saturates at about 4800 events/spill. When running with a two-processor system (i.e. we re-distribute the DAQ-unit over two RAID 8235 cards) we could reach more than 5000 events per spill; at that point we were limited by the buffer space in the VME/HiPPI modules.

During the same beam period the TRT collaboration ran routinely (one TRT sector, plus 200 words of beam chambers data read on an event by event basis) reporting read-out rates of about 4000 events per spill. The CAMAC read-out time was here the limiting factor.

1. When running in real conditions (i.e. with detector data in addition to the header), the detector buffer size might be a limiting factor.

2. We remind that the TRT uses VME to HiPPI "intelligent" interfaces with firmware providing high performance TRT sector acquisition during the spill.

Figure 23 TRT test beam read-out rates



From the above we conclude that, when considering the basic DAQ overhead, the system performs very well, also taking into account that some of the elements, such as the RAID 8235 processor, are by now almost 5 years old. The major contributions to the performance come from the quality of underlying the real-time UNIX system and from software design trade offs. For example the decision to use an existing threading system (derived from [27]) instead of the one provided by ISIS was vital. It allowed to control thread scheduling with minimal disturbance to the data flow, at the cost of some additional complication in the integration with the (ISIS based) control system.

On the side of the data transfer, program controlled read-out via the VIC has obvious performance limitations, yet it is a simple solution which has shown to be more than adequate for the test beam requirements where a few hundreds of relatively small (2-4 Kbytes) events are read-out. Performance can be improved by using DMA transfers (instead of program controlled ones) over the VIC [TN 96]. The FDL based event builder (discussed in section 3.2) is an option providing a potentially large performance improvement. It could serve the requirements of the test beam in the medium (2-3 years) term future. The RD13 DAQ supports a range of data transfer options (which, if required, can be mixed in the same configuration), the cost-effective one can be selected on the basis of the detector setup requirements. We also remark that a further degree of flexibility is given by the possibility to configure without the event builder stage in the case of a single detector (as for the TRT run discussed above).

8 Project Assessment and Future Directions

The RD13 project has been a test and learning ground in a number of areas:

- Data flow: the focus for the study and the evaluation of real-time UNIX, multiprocessor systems, shared memory systems, the application of software engineering to real-time applications.

- Control-like functions: the application of distributed computing concepts to message passing, finite state machines, hierarchical control, process management.
- Data bases: the use of commercial DBMS(-like) products as a framework for describing and parametrising the DAQ system.
- Ancillary modules: a practical ground to evaluate and apply commercial software products in several areas such as graphical user interfaces, DBMS, software development environments.
- System integration: to manage source code, to build multi-platform elements and system prototypes, to test components and full DAQ prototypes.
- Architecture studies: to project the results of RD13 into the context of an LHC-like experiment advanced components (e.g. a switch based event builder) and system level DAQ models have been produced.

This has resulted in experience with hardware and software technology and products, prototypes of advanced DAQ components and a line of complete systems, of increasing complexity and functionality, used in real life test beam runs.

The use of commercial software products (as opposed to in house developments) has always been emphasised by RD13. We think that such products will be more and more pervasive in the future. This is an important change of culture which will have to be spread to the users community: experiments will have to budget for software components as well as hardware.

Integration (of hardware and software products, different modules, etc.) has been another focus for the activity of the project. The availability of standards is the key element for success; while on the hardware side standardisation is a common practice, the same is not yet true for software. If necessary and possible, trade off between feature and easy (or possible) integration should be considered.

The multi-institute nature of RD13 is similar, although on a smaller scale, to that which the DAQ teams of LHC experiments will face. Combining developments done in geographically spread locations into a coherent system which, because of technical and financial constraints, must be located in a unique place is a challenging task. The optimization of the use of the available resources will require such an approach for LHC too and its success will depend on the existence of a formal framework to guarantee coherency and homogeneity in a geographically dispersed team with a unique integration point.

The RD13 system in its various forms (stand alone, single detector, multi-detector read-out) is now a mature and reliable product, as its continuous use in the ATLAS test beam shows. Its technical features of scalability, modularity and technology independence make its potential use wider. For example the use of the RD13 DAQ system in the full ATLAS context (i.e. its standardisation for all the ATLAS test beams) or even wider (e.g. other test beams) is mainly limited by the resources required to support the various installations. Given a suitable engineering effort, the installation in and the support of the test beams could be sub-contracted to industry. Thus releasing the unique expertise and know-how of the CERN personnel for the main objective, the development of an LHC experiment DAQ.

From the above discussion, we conclude that the RD13 project has fulfilled its objectives and has demonstrated its usefulness. It is however clear to us that we are not ready for an LHC-like DAQ system. We do have plans to continue the work initiated in RD13 and practical reasons suggest that this

is better done within an LHC collaboration. The time of the generic (experiment independent) DAQ system is well behind us, the experiment will provide a better focus and a more direct evolution path, more adequate resources will be available as well as the real working environment. This does not mean that we reject collaboration outside the experiment's frame. On the contrary we do encourage common activities with other experiments on specific, technical topics of common interest; the integration into the final DAQ system remains, and cannot be otherwise, with the experiment.

We also think that the RD13 working model, where laboratory developments are complemented by their periodic use in a real life environment, is one of the key elements of our success. The test beam runs provide real users, hence realistic feed-back, and a stringent "time to successful development" requirement, thus contributing to focus the activity of the project. We should, however, point out that, as we have experienced, the lack of resources for the real-life application (e.g. the test beam) may endanger the whole project.

Acknowledgements

We are pleased to acknowledge the fruitful collaboration with the various detector teams: RD6, RD34, RD11, RD3, MSGC. We gratefully acknowledge the support of R. McLaren (ECP/EDU) and E. Van der Bij (ECP/EDU) for the organisation of the HiPPI setup. We thank G. Kellner for contributing substantially to the purchase of the software products and tools selected by RD13. We gratefully acknowledge the professional qualities of J. Tedesco and H. Rotival (DCS company) our system managers. We gratefully acknowledge the contribution of W. Greiman (LBL) to the event builder simulation studies. We thank F. Gagliardi (CN/PDP) and B. Panzer-Steindel (CN/PDP) for contributing the central data recording system. We are also indebted to D. Klein (ECP/SA) for the precious work in the organisation and support of many aspects of the RD13 life, in particular for the documentation system.

RD13 Technical Notes

TN 0: Index of RD13 / D.Klein
TN 1: RD13 Workplan - Phase1 / L.Mapelli
TN 2: How to produce a RD13 Note / D.Klein
TN 3: RD13 Dataflow / G.Mornacchi
TN 4: RD13 Dataflow Requirements / G.Mornacchi
TN 5: Run Control for RD13 / R.Jones
TN 6: Error message Facility for RD13 / R.Jones
TN 7: Basic Libraries for IDT Monitor / S.Buono
TN 8: Some basic informations on interrupt.handling within IDT / R.Ferrari
TN 9: VME Interrupt Requests from CBD 8210 / R.Ferrari
TN10: Using Motif in RD13 / R.Jones et al.
TN11: RD13 Database Frame Work / G.Mornacchi
TN12: Basic Libraries for SVIC/ VIC Interface / G.Ambrosini
TN13: Basic Libraries for TC/ IX Environment / G.Fumagalli, C.Rondot
TN14: RD13 Prototype Minimal DAQ / RD13 Team
TN15: Data Flow Protocol Prototype / G.Mornacchi
TN16: Event and Data recording formats / M.Huet
TN17: Processes Synchronization in a VIC VME System / P.Y.Duval
TN18: Directory Organization for Production Software / G.Mornacchi, F.Tamburelli
TN19: Batch processing in the RD13 Computer System / G.Mornacchi, F.Tamburelli
TN20: RD13 Cluster Management Utilities / F.Tamburelli

TN21: Real -Time Requirements / G.Mornacchi
TN22: Real -Time UNIX (TC/IX) evaluation / L.Mapelli, G.Mornacchi, R.Jones
TN23: Using the VME trigger module through macros / C.Rondot
TN24: SVIC 7213/ VIC 8250 / G.Ambrosini
TN25: Design of a data flow protocol with StP / A.Khodabandeh
TN26: Using ISIS and META for Run Control in RD13 / R.Jones, G.Mornacchi, G.Polesello
TN27: The TC/IX process priority in pictures / C.Rondot
TN28: Tools for TC/IX / C.Rondot
TN29: The RD13 tree: make it by example / C.Rondot
TN30: Read Modify Write Functions in a Raid/Vic/Vme System / P.Y.Duval
TN31: Diagnostic for the RD13 System / S.Buono, D.Prigent
TN32: Data recording / M.Huet
TN33: A proposal to organise code reviews / G.Mornacchi
TN34: TCIX Systems Status / G.Mornacchi
TN35: RD13 Data Base Software / F.Tamburelli
TN36: Use of QUID as Data Base Framework / G.Mornacchi
TN37: Use of RD13 Problems Data Base / F.Tamburelli
TN38: About the VMV BUS / D.Prigent
TN39: Trigger Tools / C.Rondot
TN40: Status Report of RD13 (1992) / L.Mapelli et al.
TN41: A hardware setup for the SiTP test read-out / S.Buono,A.Ferrari, D.Prigent
TN42: RCL and DFE modules integration / P.Y.Duval
TN43: db_select / A.E.Topper
TN44: The RD13 DAQ system (User's Guide) / S.Buono
TN45: A library for the RAID ZCIO Timers / S.Buono
TN46: Event Monitoring in the RD13 Minimal System / G.Ambrosin,G.Mornacchi
TN47: How to build a Monitoring task in the RD13 System / G.Ambrosini
TN48: RD13 DFP Upgrade / L.Mapelli, G.Mornacchi
TN49: Extending PDFP to a multiprocessor configuration / G.Ambrosini,G.Mornacchi
TN50: Performance Study of the Artifex based DFP / A.Khodabandeh, G.Mornacchi
TN51: Users's Guide to the Artifex DAQ in the RD13 Laboratory / A.Khodabandeh,G.Fumagalli
TN52: SYSlib - platform independent system library / G.Mornacchi
TN53: A Proposal for an Asynchronous Message Interface / R.Jones,L.Mapelli,G.Mornacchi
TN54: Introduction to Petri Nets / A.Khodabandeh
TN55: Introduction to Artifex and its Petri Nets / A.Khodabandeh
TN56: RD13 System_Lab / D.Prigent
TN57: Known Problems in Using Artifex / A.Khodabandeh
TN58: RD13 Source Code Management / G.Fumagalli,P.Pinassseau
TN59: User's libraries for the RIO/Hippi 8262/S module / S.Buono,J.Hansen
TN60: Run Control Requirements / R.Jones et al.
TN61: Software Quality in RD13 / R.Jones
TN62: Testing the interface between Artifex and non-Artifex applications / R.Jones,E.Sanchez-Corral
TN63: Design of a Data Flow Protocol with Artifex / A.Khodabandeh
TN64: RD2 / RD13 DAQ system modelling / I.Gaponenko
TN65: OODBMS evaluation for a DAQ System / M.Skiadelli
TN66: Applicability for the OO technology on the RD13 databases / M.Skiadelli
TN67: The Hardware configuration database / M. Skiadelli
TN68: Evaluation of Histogramming Packages / P. Ganev, S.Hellman, R.Jones
TN69: RD13 Run Control Parameter Setting / D. Ferrato, G. Polesello
TN70: Studies on RIO/HiPPI based Event Building / S. Buono, J. Hansen, H. van de Bij
TN71: Os4 Mod / I. Gaponenko
TN72: Simulation code with MODSIM II / I.Gaponenko
TN73: Asynchronous Message Interface / G.Mornacchi
TN74: Database system for PDFP and RCL / Z.Qian

TN75: RD13 implementation of CERN Standard DMA Library / S. Buono
TN76: RD13 DAQ Builder / G.Fumagalli
TN77: ITASCA & O2: a comparison / M.Skiadelli
TN78: Status Report of RD13 / 1993 / L.Mapelli et al.
TN79: A memory management for the RIO/HiPPI 8262/S module / S.Buono
TN80: Users's libraries for the RIO/HiPPI 8262/D module / S.Buono
TN81: Run Parameters Data Base / G.Mornacchi
TN82: Managing Multiple Interrupts in EP/LX / G.Mornacchi
TN83: Evaluation of the CES HiPPI interfaces into the RD13 system / S.Buono
TN84: RD13 Basic Libraries / G.Ambrosini, G.Mornacchi
TN85: Using Quid for the implementation of the HardWare database / M.Skiadelli
TN86: Cebrax: a recording package / M.Huet
TN87: LabView Tools Manual / C.Rondot, D.Prigent
TN88: Diagnostic with LabView / C.Rondot
TN89: Requirements of a Simulation Program for DAQ Modelling / R.Spiwoks
TN90: Proposal of a Lib. and a Skeleton Prog. for DAQ Simulations / R.Spiwoks
TN91: RD13 System_Lab / D.Prigent
TN92: Interpreteur Interactif d'expressions logiques / R.Nacasch
TN93: RD13 System_Lab Vic Bus v.2 / D.Prigent
TN94: New run control library users guide / P.Y.Duval, A.Levansuu
TN95: Proposal requirements for resource manager library / R.Jones, A.Levansuu
TN96: Memory transfer tests using DMA libraries / S.Buono, D.Prigent
TN97: Modelling of the RD6 Testbeam Setup / R.Spiwoks
TN98: XRunner Evaluation Report / R.Jones
TN99: T Evaluation Report / R.Jones
TN100: RD13 Artis Installation / G.Mornacchi, E.Sanchez-Corral
TN101: Technical Guide of the Artifex based RD13 Run Control / E.Sanchez-Corral
TN102: RUNCO_SYS / E.Sanchez-Corral
TN103: Improvements in the Artifex based RD13 Run Control System / R.Jones, E.Sanchez Corral
TN104: Online Help facility for RD13 DAQ / R.Jones
TN105: Read-out module specifications for the June 1994 testbeam / S.Buono
TN106: Specifications for the RD13 hardware configurator / S.Buono
TN107: A general graphical User Interface with MODSIM II / K.Djidi
TN108: Object oriented database system evaluation for DAQ system / M.Skiadelli
TN109: Event Format Library / G.Ambrosini
TN110: Event Format User Interface / G.Ambrosini
TN111: Requirements of an Event Building System / R.Spiwoks
TN112: Run and Detector Parameters Data Base / G.Mornacchi
TN113: Local Memory Support in the RD13 DAQ / G.Mornacchi
TN114: Browser for Run Control Error Messages / I.Soloviev
TN115: Proposal on Modelling of ATLAS DAQ Architecture / I.Gaponenko, V.Kozlov
TN116: Modelling of local/global architecture at the LHC experiment / Z.Hajduk
TN117: Status Report of RD13 / 1994 / L.Mapelli et al.
TN118: A DAQ User Interface for ATLAS Simulation / K.Djidi
TN119: DAQ Simulation Library (DSL) -A Reference Manual / K.Djidi, M.Huet, R.Spiwoks
TN120: Functional Simulation of Read-Out Parts of a DAQ Architecture / V.Kozlov
TN121: A maintenance guide to the hardware database / S.Buono
TN122: TEST BEAM 94 / D.Prigent
TN123: Lab View with Test Beam - 94 / D.Prigent
TN124: The ATLAS H8 Test Beam Data Acquisition / C.Bee et al.
TN125: The trigger for the combined ATLAS Test Beam / G.Polesello
TN126: OS Event Buffer Support in ATLAS Testbeam DAQ / A.Miotto
TN127: New Run Parameter Database Library / I.Soloviev
TN128: Process Manager for RD13 / D.Ferrato

TN129: The HiPPI/D firmware for the RD13 EB System Prototype / R.Spiwoks
TN130: User Library for the RD13 DstFmw / R.Spiwoks
TN131: Resource manager Library / R.Jones, A.Levansuu
TN132: Design and Simulation of Fibre Channel Based Event Builders / W.Greiman
TN133: OMW / Kappa CASE tool Overview/ R.Jones
TN134: Using ITASCA for EMF messages logging and browsing/ I.Soloviev
TN135: HWdb & SWdb in ITASCA / Z.Qian
TN136: Global Architecture for the ATLAS DAQ and Trigger / L.Mapelli
TN137: EmfDB Programmers Guide / R.Jones
TN138: Fast DATA Link / S.Eshghi
TN139: XDesigner Exercise / R.Jones
TN140: OMW Exercise / R.Jones
TN141: Online Volume Bookkeeping Requirements / C.Maidantchick, R.Jones
TN142: Event Builder Application using the FDL / S.Eshghi
TN143: The RD13 HiPPI/S firmware / R.Spiwoks
TN144: User Library for the HiPPI/S module / R.Spiwoks
TN145: Tape Labelling in RD13 DAQ / R.Jones
TN146: User Dialogues in RD13 DAQ / R.Jones
TN147: Installing a new Raid 8235 / EP-LX / J.Tedesco
TN148: Event checker / M.Niculescu
TN149: Databases uses in RD13 / P.Y.Duval
TN150: PDFP db performance measurement / Z.Qian
TN151: Use of ITASCA status report / P.Y.Duval, Z.Qian
TN152: Data Acquisition General Control User Interface / M. Caprini
TN153: An approach to the object description of the DAQ system / Y.Ryabov, I.Soloviev
TN154: Using ITASCA for Run Control & Detector Parameters data base/ I.Soloviev
TN155: Hardware data base editor, programmer's guide / A. Patel
TN156: Level-3 Program Parametrisation / G. Fumagalli, G. Mornacchi

RD13 Publications

R. Jones et al., Using Motif in RD-13, Proceedings of Motif'91 Conference, Washington, USA, 1991, CERN/ECP 92-11.

R. Jones et al., Using ISIS and META for Run-Control in RD13, Proc.Intern.Workshop Softw.Eng, 1992, pp. 199-202.

R. Jones et al., Using Motif in RD-13, Proc.Intern.Workshop Softw.Eng, 1992, pp. 149-156.

L. Mapelli, Software for future Data Acquisition - A Chance for Real-Time UNIX?, Proceedings of CHEP'92, 1992, CERN92-07, pp. 60-68.

G. Mornacchi et al., The RD13 Scalable Data Acquisition System, Proceedings of CHEP'92, 1992, CERN92-07, pp. 281-284.

R. Jones et al., Building Distributed Run-control in UNIX, Proceedings of CHEP'92, 1992, CERN92-07, pp. 289-292.

C. Rondot et al., Graphical user interfaces for a Data Acquisition System, Proceedings of CHEP'92, 1992, CERN92-07, pp. 432-438.

A. Khodabandeh et al., Use of CASE tools in the design of a data flow protocol for the RD13 data acquisition system, Proceedings of CHEP'92, 1992, CERN92-07, pp. 583-588.

G. Fumagalli et al., Use of Real-Time UNIX in data acquisition for HEP, Proceedings of CHEP'92, 1992, CERN92-07, pp. 632-638.

A.Topper, Simulation of High Performance DAQ System, thesis for Norwegian Institute of Technology, May 1993.

M. Aguer et al., Software Engineering Techniques and CASE Tools in RD13, Proceedings of AIHEP'93,World Scientific, pp. 29-38.

S. Buono et al., A Hierarchical and Distributed Control Architecture for HEP Big Data Acquisition Systems, Proceedings of AIHEP'93,World Scientific, pp. 77-82.

G. Ambrosini et al., Software Engineering Techniques and CASE Tools in RD13, Proceedings of ICALEPCS'93, North-Holland, pp 383-386.

G. Ambrosini et al., Real-Time UNIX in HEP Data Acquisition, Proceedings of ICALEPCS'93, North-Holland, pp 213-216.

M. Aguer et al., The RD13 Data Acquisition system, Proceedings of CHEP'94, LBL-35822, pp 104-108.

G. Ambrosini et al., Modelling of Data Acquisition Systems, Proceedings of CHEP'94, LBL-35822, pp 109-113.

G. Ambrosini et al., OODBMS for a DAQ system, Proceedings of CHEP'94, LBL-35822, pp 143-146.

M. Skiadelli, Object oriented database system evaluation for the DAQ system, thesis for Computing Science degree from Patras Polytechnic School Greece, March 1994.

G. Ambrosini et al., Application of OO methodology and CASE to a DAQ system, Proceedings of CHEP'95, to be published.

G. Ambrosini et al., Studies of switch-based event building systems in RD13, Proceedings of CHEP'95, to be published.

G. Ambrosini et al., Experience using a distributed object oriented data base for a DAQ system, Proceedings of CHEP'95, to be published.

L. Mapelli et al., The DAQ and trigger system of the ATLAS experiment at the LHC, Proceedings of CHEP'95, to be published.

9 References

[1] L.Mapelli et al., A Scalable Data Taking System at a Test Beam for LHC, CERN/DRDC/90-64/P16, CERN/DRDC/90-64/P16 Add.1, CERN/DRDC/90-64/P16 Add.2 (1990).

[2] L.Mapelli et al., Status Report of A Scalable Data Taking System at a Test Beam for LHC, CERN/DRDC 92-13.

[3] L.Mapelli et al., Status Report of A Scalable Data Taking System at a Test Beam for LHC, CERN/DRDC 94-24.

[4] Creative Electronics Systems S.A., FDL 8050 User's Manual, Version 1.0, DOC 8050/UM, 1995.

[5] Creative Electronics Systems S.A., VIC 8251/F User's manual, 1993.

[6] B. Panzer-Steindel, private communication.

[7] KLC group Inc., XRT Builder Guide & Reference Manual, Ref No. BLGDE-GRAPH/M/240-07/94.

[8] B.Dolgoshein et al., Status Report of Integrated high-rate transition radiation detector and tracking chamber for the LHC, CERN/DRDC93-46.

[9] M.Cavalli-Sforza and, M.Nessi et al., Status Report of Developments for a scintillator tile sampling hadron calorimeter with "longitudinal" tile configuration, CERN/DRDC/92-48.

[10] . Weilhammer, G. Hall et al., Development of High Resolution Si Strip Detectors for Experiments at High Luminosity at the LHC, CERN/DRDC/93-30.

[11] R.K. Bock et al., Status Report of Embedded Architectures for Second-level Triggering in LHC Experiments, CERN/DRDC/93-12 / RD-11.

[12] Creative Electronics Systems S.A., HIPPI 8262/D VME to HIIPI Destination Interface User's Manual, 1992.

[13] Creative Electronics Systems S.A., RAID 8235 VME RISC Processor Board User's Manual,1992.

[14] Creative Electronics Systems S.A., RCB 8047 CORBO VME Read-Out Control Board User's Manual, 1992.

[15] K.P.Birman et al., The ISIS SYSTEM MANUAL, Version 2.1.

[16] Quid version 2.0 User Manual, Artis srl, November 1992.

[17] CentreLine Software, Inc., Cambridge Massachusetts, U.S.A, 1990.

[18] FRAME MAKER, International Version 3.0, FRAME Technology 1991.

[19] Imperial Software Technology. X-Designer User Manual.

[20] DataViews Graph Widgets Programmer's Manual 2.0, V.I. Corp. 1993. Doc # 2.0DVGW

[21] LabVIEW 2 User Manual, National Instruments Corp., 1990.

[22] Control Data, TC/IX Users's Guide, CDC May 1991.

[23] R.Bauer, A review of LynxOS, Unix Review, September 1990.

[24] IEEE Std POSIX 1003.1-1988

[25] IEEE Std POSIX 1003.4-1991

[26] Creative Electronics Systems S.A., FIC 8234 Dual 68040 Fast Intelligent Controller User's manual, 1992.

[27] R.Russel, G.Mornacchi, VOS a virtual operating system, CERN July 1990.

[28] Creative Electronics Systems S.A., VCC 2117 Intelligent CAMAC Crate Controller User's Manual, 1992.

[33] B. Thiercelin, FDL 8050 Evaluation, Version 1.0, ECP/ESS internal document.

[34] E.M. Rimmer, J. Ogilvie, On-line Support for Labelled Magnetic Tapes, CERN, 10 March 1977 revised November 3 1981.

[35] John E. Davis, private communication.

[36] R. Jones, http://rd13doc.cern.ch/onlineHelp/eventDump.html

[37] L. Mapelli, The Challenge of triggering and data acquisition at supercollider experiments, Nuc. Inst. & Meth. Phys. Research A315 (1992) 460-471.

[38] Creative Electronics Systems S.A., RIO 8260 Processor User's Manual version 1.0, May 1992.

[39] The ATLAS collaboration, Technical Proposal for a general Purpose Experiment at the large Hadron Collider at CERN, CERN/LHCC/94-43.

[40] S. Hunt et al., SIMDAQ - A System for Modelling DAQ/Trigger Systems, Proceedings IEEE RT95 Conference, to be published in IEEE Trans. on Nuclear Science.

[41] L. Kleinrock, Queuing Systems, Vol. 1, Wiley, 1975.

[42] G. Mornacchi, ATLAS internal note DAQ-NO-23, 1994.

[43] M.D. Hill et al., Wisconsin Architectural Research Tool Set, Comp. Science Department, University of Wisconsin, 1993.

[44] ARTIFEX, Artifex Environment User Guide, ARTIS 1991.

[45] Intellicorp, Inc., Object Management Workbench User's Guide Version 1.0, Pub # OMW1.0-UG-2,1994.

[46] P. Butterworth, A. Otis, J. Stein, The Gemstone Object Database Management System, Communications of the ACM, October 1991, Vol.34, No. 10.

[47] ITASCA Systems, Inc., ITASCA Distributed Object Database Management System, Technical Summary R2.1,1992.

[48] VERILOG S.A., Logiscope Editor 3.3 Reference Manual, D/LEXX/RA/330/334, 1993.

[49] B. Belliner, CVS II:parallelizing software development, Proceedings of the Winter 1990 USENIX Conference, Washington, DC, January 1990. USENIX, 1990.

[50] T.Berners-Lee et al. World Wide Web; An Architecture for Wide-Area Hypertext. CERN, 1991.

[51] K. Bos et al., Object oriented approach to software development for LHC experiments, CERN/DRDC/94-9/P55.

[52] The PASS Collaboration, The PASS Project Architecture Model, 1994.

[53] G.Stefanini et al., Status Report of Optoelectronic analogue signal transfer for LHC detectors, CERN/DRDC/93-35.

[54] J. Martin and J.J. Odell, Object Oriented Methods A Foundation, Prentice Hall, 1995.

[55] Intellicorp, Inc., Kappa User's Guide Version 3.0, Pub. # K3.0-UG-2, 1993.

[56] Object Management Group, The OMG Object Model, Document 91.9.1, 1993.

[57] M. Skiadelli, Object oriented database system evaluation for the DAQ system, thesis for Computing Science degree from Patras Polytechnic School Greece, March 1994.

[58] HiPPI Standard, ANSI X3T9.3/91-005.

[59] Input Output Systems Corporation, IOSC HiPPI Switch, 1994.

[60] Creative Electronics Systems S.A., RIO 8260 RISC I/O Processor, User's Manual, 1992.

[61] Creative Electronics Systems S.A., HiPPI 8262/S VME to HiPPI Source Interface, User's Manual, 1992.

[62] Creative Electronics Systems S.A., HiPPI 8262/D VME to HiPPI Destination Interface, User's Manual, 1992.

[63] Creative Electronics Systems S.A., RAID 8239 VME RISC Processor Board, User's Manual, 1992

[64] CACI Products Co., MODSIM II, The Language for Object-Oriented Programming, Reference Manual, 1991.

[65] J. Baines et al., ATRIG 1.00, ATLAS Trigger Simulation User Guide, 1994.

[66] I. Mandjavidze, Review of ATM, FibreChannel and Conical Network Simulations, Proc. of Int. Conf. on DAQ Systems, FNAL, Batavia, Illinois, USA, 1994.