EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

CERN/DRDC/92-13
RD13/Status Report
13 March 1992

Status Report of

A SCALABLE DATA TAKING SYSTEM AT A TEST BEAM FOR LHC

C.P. Bee, R. Bonino[1], S. Buono, P. Ganev, S. Hellman, R. Jones, A. Khodabandeh, L. Mapelli[2], G. Mornacchi, D. Prigent, S. Stapnes, F. Tamburelli, N. Zaganidis
CERN, Geneva, Switzerland

P.Y. Duval, F. Etienne, D. Ferrato, A. Le Van Suu, R. Nacasch, C. Rondot[3], S. Tisserant
Centre de Physique des Particules de Marseille, IN2P3, France

G. Ambrosini, R. Ferrari, G. Fumagalli, A. Lanza, F. Pastore, G. Polesello
Dipartimento di Fisica dell'Universita' e Sezione INFN di Pavia, Italy

M.L. Ferrer, G. Mirabelli
INFN - Lab. Naz. di Frascati / Dip. di Fisica dell'Universita' e Sez. INFN di Roma, Italy

M. Aguer, T. Choulette, M. Coret, J.L. Fallou, M. Huet, J. Pain
Departement de Physique Nucleaire - STEN, C.E. Saclay, France

1. Introduction

The RD13 project was approved in April 1991 for the development of a scalable data taking system suitable to host various LHC studies [1]. The basic motivations come from the conviction that, being too early for a 'top-down' design of a full DAQ architecture, a lot can be gained from the study of elements of a readout, triggering and data acquisition and their integration into a fully functional system.

.his document is organised in three parts, preceded (sect. 2) by a summary of the motivations which have inspired the RD13 project and the consequent goals which we identified in the proposal:

1. detailed description of the activities promoted to satisfy the goals which have been the guiding principles of the project in the first phase, namely
   - construction of a scalable DAQ framework (sect. 3)
   - evaluation of the use of a Real-Time UNIX operating system (sect. 4)
   - commercial products: for system controls (sect. 5), for interfaces and documentation (sect. 6)
   - software engineering (sect. 7)
2. status of the development of the RD13 data acquisition system (sect. 8)
3. proposal for the next phase of the project (sect. 9).

1. Now at the University of Geneva
2. Spokesperson
3. Presently at CERN

The report is concluded, in section 10, by an account of the necessary resources for the continuation of the project. The original RD13 team has expanded to include one new direct collaborator (IN2P3 Marseille) and contributions from the CERN ECP/PT group.

2. Motivations and goals of the project

The time scale for LHC experimentation and the inadequacy of the existing readout and DAQ technology make a 'top-down' design premature. A more appropriate preparation for LHC is to spend some time and resources in investigating system components and system integration aspects. The investigation is more effective if done in a realistic environment, such as the data taking phase of a detector prototype at a test beam. Such a setup has the double advantage of serving the increasing data taking demands from the evolution of the detector readout and triggering electronics on one hand and, at the same time, helping the smooth evolution of the DAQ system by forcing a continuous integration of newly developed components into a working environment.

A further motivation drives RD13, the conviction that the traditional standard High Energy Physics methods for online software developments would fail, with the obvious disastrous consequences for LHC experimentation, given the undeniable complexity of the required systems. Much has to be done to find suitable methods for complex online system designs and much has to be learned in the area of software engineering tools. The ability to be constantly evaluating and experiencing in an environment close to the real one, is the great advantage of a 'down-to-the-earth' learning ground, as proposed by the RD13 collaboration.

To solve the problems which have motivated our project, four major goals were envisaged in the RD13 proposal and have constituted the working plan of the RD13 collaboration in this first phase.

1. The core of the project is the construction of a *DAQ framework* which satisfies requirements of *scalability*, in both number of data sources and performance (processing power and bandwidth); *modularity*, i.e. partitioned in functional units; *openness*, for a smooth integration of new components and extra features. It is necessary to point out that while such features are easily implemented in a well designed hardware architecture, they constitute a big challenge for the design of the software layout. This is our first goal.
2. Such a DAQ framework is an ideal environment for pursuing specific DAQ R&D activities, following a 'bottom-up' approach, for the clear identification and development of the building blocks suitable for the full system design. The R&D activity which we indicated in the proposal as fundamental in the first phase of the project is the investigation of the use of *Real-Time UNIX operating systems* in RISC-based front-end processors, to assess their combined potential, to converge towards operating system standards and to reach a full platform independence.
3. The control of all the aspects of a sophisticated DAQ system demands specialised software to complement UNIX with additional services and facilities. The use of a powerful tool-kit for distributed programming is described in section 5. Other aspects of DAQ R&D, more directly dependent on a modular architecture, involve the integration of suitable *commercial products*.
4. The last project goal is the exploitation of *software engineering* techniques, in order to indicate their overall suitability for DAQ software design and implementation and to acquire expertise in a technology considered very powerful for complex software development but with which the HEP community remains unfamiliar.

3. Construction of the scalable DAQ framework

The guiding principle of an evolutive (i.e. scalable, modular and open) data acquisition system led us to the architecture which we outline in this section. Being the skeleton of our project, we decided to present the DAQ framework in all its fundamental aspects:

- the basic concept of the RD13 DAQ (sect. 3.1)
- the hardware (3.2) and the software (3.3) layouts supporting scalability
- the data flow protocol (3.4) designed to meet the modularity requirement
- the data base (3.5) and distributed computing (3.6) aspects.

3.1 Conceptual description

We define two main functional components in the DAQ: the back-end and the front-end. The former fulfills the role of the development environment, control management, high-level monitoring and user interface. The latter constitutes the DAQ environment proper, whose main functions are data readout and distribution, low level monitoring and data recording.

The main element of the back-end environment is the workstation, providing users with the interactive environment to control and monitor the data acquisition.

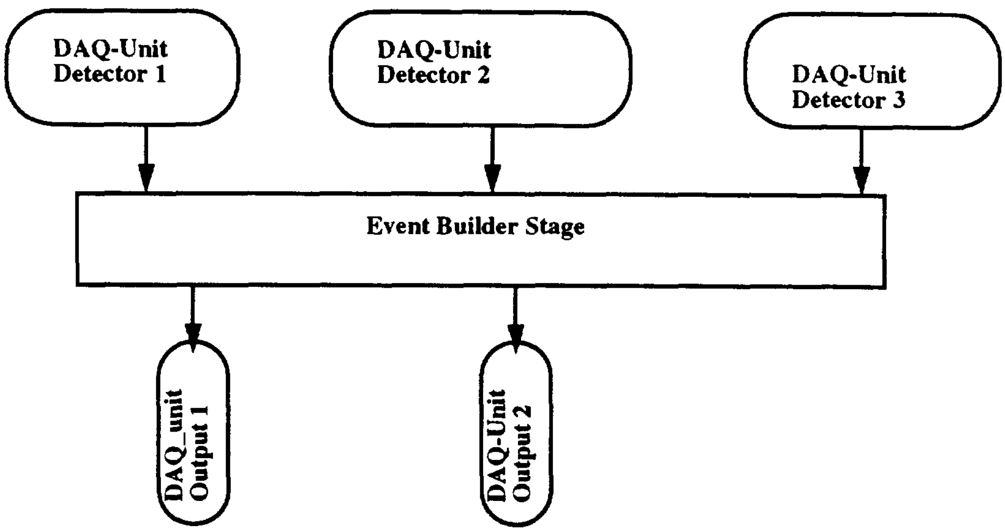
In the front-end we identify the following modular elements:

- *The DAQ-Unit.* It is in itself a local DAQ which consists of an input data path, a data processing part and an output data path. The hardware configuration reading a sub-detector, performing some data formatting or filtering and feeding the resulting data into an event builder would be an example of DAQ-Unit. Processors, input and output cards are the hardware building blocks of a DAQ-Unit. An integration element (e.g. a backplane bus) is needed to group the hardware components inside a DAQ-Unit. We require that all components (I/O modules and processors) within a DAQ-Unit share memory space so that only pointers to events need to be moved. From the point of view of the data flow this is the main characteristic of a DAQ-Unit.
- *The DAQ-Module.* A software program running on the processors and providing a data acquisition function (read-out, recording, etc.). A DAQ-Module has an input and an output part connecting respectively to the output and input parts of another module or another DAQ-Unit. A protocol ensuring the management of the connections and the transfer of data is used to make this interconnection process, and to make the data transport mechanism transparent to the software.

A complete data acquisition system results from the interconnection of one or more DAQ-Units into a network where data flow from the front-end electronics to the recording media (an example is shown in figure 1). The connection between DAQ-Units is implemented by the input/output modules and may run on different data transport media (a point-to-point link such as HiPPI is a relevant example). The I/O modules themselves are of course dependent on the transport medium, our only requirement being that they integrate with the processors in the DAQ-Unit.

Integration of back-end and front-end is achieved primarily via a local area network (e.g. ethernet or FDDI), interconnecting all back-end workstations and front-end processors, for the distribution of the information necessary to control and manage the system. Back-end workstations may also be integrated in a DAQ-Unit if a suitable means to share memory between the workstation and the front-end modules is available.

FIGURE 1. DAQ configuration example.



Scalability (upward, e.g. more data sources, or downward, e.g. less data sources) is a consequence of the partitioning of the system into modular, interconnected components which can be combined to build different data acquisition systems. Scalability in processing power can be achieved by adding more processing elements to a DAQ-Unit, hence redistributing its functions on more processors.

3.2 Hardware layout

The RD13 hardware layout matches the general architecture described in the previous section. Our choices have also been guided by the principle of adhering to industry standards and tracking their evolution.

The system is currently based on VMEbus, using the VICbus (Vertical Inter-Crate), to link VME crates and to integrate back-end workstations with the front-end via the SVIC (Sbus to VIC interface). The processing element we have selected is the MIPS 3000 based RAID 8235 which responds to both our requirements for a front-end processor and to our intention to evaluate a RISC architecture in the front-end. VIC modules are our first examples of I/O elements.

We are firmly on the industrial evolutionary path: we can easily incorporate VME-64, and we are architecturally compatible with future backplanes such as Futurebus+. The layout is able to accept the RIO 8260 based HiPPI interface or a new processor card based on a combination of the MIPS 3000 and 4000 chips (see section 9).

3.3 Software layout

The software layout supports the framework outlined in section 3.1. We have identified three broad areas necessary to complement the hardware layout:

- the dataflow protocol, to allow a network-like composition of DAQ-Units and DAQ-Modules, and transfer of data within the network independently of the interconnection medium,
- the support for distributed computing, for the coherent organisation of the system elements,
- the database framework, to have a common approach to the definition and implementation of databases for the management of all the system components,

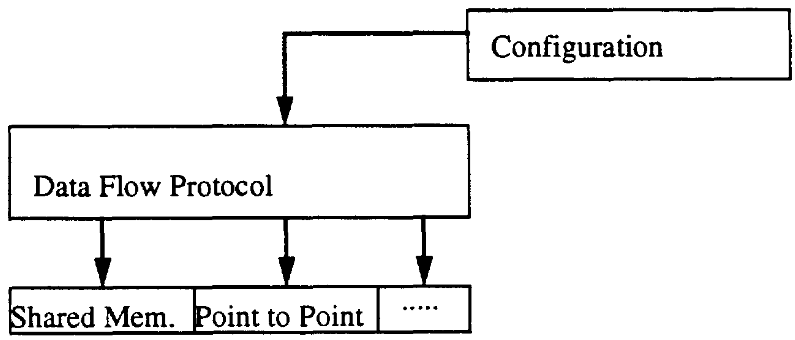
In addition, UNIX, as the common operating system, C as the programming language, NFS (Network File System) as the means of file sharing and TCP/IP as the common protocol for communication are the low level tools for software development.

3.4 Data flow protocol

The concepts outlined in section 3.1 and the availability of shared memory suggest the need for a common set of rules to define, configure and transport data. In our terminology, this is called the data flow protocol (DFP). RD13 Technical Notes [TN 3] and [TN 4] define the protocol and its requirements respectively. The DFP consists of two main parts (figure 2).

- Configuration management: a description of the DAQ in terms of DAQ-Units, modules and their interconnection coupled with protocol primitives to create the network of DAQ-Units at start-up.
- Data transport management: a two layer protocol to move data between DAQ-Units or modules. The upper layer is independent of the medium used to transport the event, the lower layer implements the transfer and the synchronisation proper of the physical media (e.g. shared memory inside a DAQ-Unit, point to point link, etc.).

FIGURE 2. Data Flow Protocol



3.5 Data base framework

The data acquisition system involves a large number of parameters to describe its hardware and software components, which need to be accessed by different parts of the DAQ. Parameters are grouped in different data blocks associated to different DAQ components (e.g. both the run control and the data flow have their own set of parameters) and are specified and implemented by different people. Each of these blocks of data, related to a system component, is what we call a Data Base. We need to describe in a complete and consistent way and access in a common and efficient way the parameters and their inter-relationships.

The data base framework gives this common approach to the definition, implementation and access to data bases. It provides functions for data modelling and data storage, Real-Time access to the data, tools to produce libraries for run-time access to the data and interactive data browsers. Its implementation will be based on commercial products and we expect a minimal development effort [TN 11].

A combination of a CASE tool such as StP and a relational data base management system such as ORACLE has provided a preliminary implementation of the framework. We intend to pursue investigations into object oriented data base management systems (OODBMS) as commercial products based on this technology seem, in fact, to fulfill our requirements.

3.6 Distributed computing

Sophisticated distributed DAQ systems, such as the RD13 DAQ, demand specialised software in order to control all the aspects of the data acquisition system. Processes need to cooperate to perform processing functions, and to share data and information. It must be possible to synchronise processes and monitor their progress. UNIX does not provide all the services and facilities required to implement such applications and so we have looked elsewhere to find tools that can provide the missing features.

One such tool is ISIS [2], a toolkit for distributed and fault-tolerant programming. ISIS started life as a research project at Cornell University and has since become a commercial product distributed by ISIS Distributed Systems Inc. The toolkit is a set of fault-tolerant software protocols that are accessed by an application programmer interface. Included is support for groups of cooperating processes, replicated data, distributed computation and fault tolerance.

The toolkit embodies a simple yet powerful metaphor for network computing called *virtual synchrony* by which complex distributed events appear to occur one-at-a-time, in *synchrony*. In reality many events are happening concurrently, exploiting the parallelism of the distributed computing environment.

We have found ISIS to be a reliable and robust product that simplifies interprocess communication. Its notion of location independence removes an obstacle for distributing applications. Its adherence to standard UNIX facilities has allowed us to break-down the traditional barrier between the back-end workstations and the front-end embedded electronic systems of a data acquisition system. A detailed description of ISIS and its features goes beyond the scope of this report and can be found in a comprehensive RD13 internal note [TN 26].

4. Real-Time UNIX operating system

The selection and evaluation, in a realistic application environment, of a Real-Time UNIX operating system is one of the main topics for study that we have identified in the proposal.

The evaluation has proceeded in two directions:

- the system features have been extensively tested against our requirements [TN 21]; the results are summarised in section 4.2.1
- a performance study of the selected operating system has been carried out, particularly for its Real-Time features.

A full account of the evaluation procedure can be found in [TN 22].

Our requirements for a Real-Time UNIX operating system are summarised in the following points:

- *UNIX compatibility.* The Real-Time UNIX should conform to the current main UNIX flavours (i.e. System V and BSD), which constitute the UNIX environment that is found on most workstations. We also require that the evolution path of the selected Real-Time UNIX tracks the operating system standardisation efforts (e.g. POSIX).
- *Real-Time features.* Data acquisition applications require system facilities not present in the original UNIX. In particular, a pre-emptable kernel, a priority based pre-emptive scheduler, efficient inter process communication facilities, easy and efficient access to the I/O bus (e.g. VME and VSB).
- *Support for multi-processors.* We aim at a system which will evolve to include many processors in the front-end. These processors will have to cooperate in a coherent way: some will be grouped into a DAQ-Unit, while functional units are geographically distributed in the overall system. The Real-Time UNIX should support this model of multi-processing as a basic requirement for the scalability of the system.

4.1 The TC/IX Real-Time UNIX

A survey of the Real-Time UNIX market led us to conclude that LynxOS [3] is currently the best Real-Time UNIX kernel for our purposes. This conclusion has been further strengthened by the fact that LynxOS has become a CERN choice in the accelerator divisions. TC/IX [4] is the port to the MIPS architecture of the LynxOS kernel marketed by CDC. In short TC/IX includes:

- Real-Time kernel, the MIPS port of LynxOS 1.2
- complete UNIX system fully compatible with both System V release 3 and BSD 4.3
- additional Real-Time features not found in LynxOS, in particular access to the VME/VSB buses and to the RAID card hardware facilities.

TC/IX also implements a scalable multi-processor model where multiple CPUs cooperate communicating via VME/VSB, while maintaining their own copy of the kernel. This is achieved by the extension of inter-process communication primitives to the cluster of processors.

4.2 TC/IX evaluation

4.2.1 Compliance to RD13 requirements

Our extensive evaluation of TC/IX has convinced us of its good match with our requirements:

- *UNIX compatibility.* We have, effortlessly, ported to TC/IX a number of large UNIX "intensive" applications including ISIS [2], Vos (a heavy user of UNIX system calls) [5], Motif applications produced with XDesigner, the Network Queueing System (NQS) [6] and all the major CERN libraries. An existing data acquisition system, originally developed for the WA89 experiment, has also been ported to TC/IX without modifications. In addition we also run GEANT based Monte-carlos on the RD13 NQS cluster which spans SUN workstations and RAID processors. We conclude that TC/IX, from the "pure" UNIX point of view, is on the same level as the UNIXes normally found on workstations.
- *Real-Time features.* TC/IX provides the basic features to make UNIX a Real-Time system and adds facilities to user programs, normally available only to device drivers (physical shared memory, mapping of VME/VSB devices, connection to interrupts, kernel semaphores). External events, such as interrupts, devices and internal resources can easily and efficiently be exploited. The main feature that TC/IX currently lacks is the support for user mode threads to efficiently implement multi-tasking within a single unit of resources, i.e. a process. This facility, as well as the other POSIX 1003.4 [7] features, are expected to be available later in the year.
- *Support for multi-processors.* Although this feature is available in alpha test form, we have not tested it yet since our initial minimal system is single-processor. The evaluation of the multi-processing capabilities of TC/IX is part of our second year program.

4.2.2 TC/IX performance analysis

Our performance analysis was made in two directions:

- *Single Real-Time feature benchmarks.* We have made measurements for
  - interrupt latency, the delay to activate the first instruction in the interrupt handler following the external interrupt signal
  - task dispatching, the time to pass the control from the interrupt service routine to the user-mode code
  - semaphore handling, the overhead to flip a semaphore
  - context switching, the time to switch the CPU from one process to another
  - I/O operations.

The measurements have been done under different CPU load conditions:

A. CPU free, i.e. no other application was competing for CPU time
B. CPU loaded with a CPU-bound application (Whetstone benchmark) running at a lower priority than the measurement task
C. CPU loaded with the Whetstone benchmark running at the same priority.

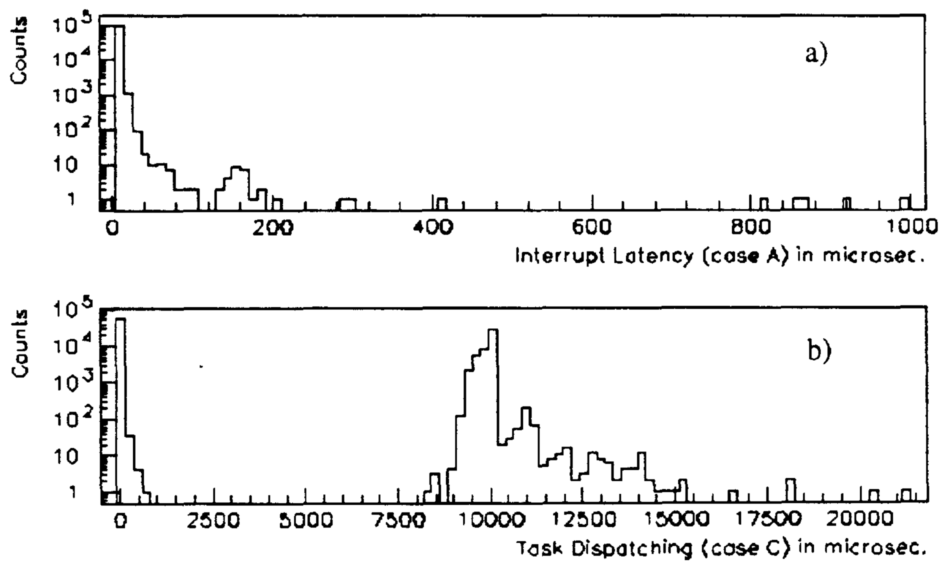
Table 1: Single Real-Time feature benchmarks[a]

	A		B		C	
	mean	rms	mean	rms	mean	rms[b]
Interrupt latency	10	2	15	2	27	16
Task dispatching	17	4	27	1	8.4 ms	1.2 ms
TC/IX semaphores	8	1	8	1	17	75
SysV semaphores	33	4	34	2	65	127
Context switching	13		n/a[c]		n/a	
I/O: Camac read	2.5		n/a		n/a	
I/O: Camac CSSA	6.5		n/a		n/a	
I/O: VME read	0.2		n/a		n/a	

a. Default time units are µs, unless otherwise stated. All value have been corrected to take into account the overhead of the measuring procedure (i.e. 2.5 µs of Camac access).
b. The reason for the large rms values is apparent in figure 3b, which is an example of the typical 'fair sharing' of CPU resources amongst tasks of equal priority.
c. not/applicable.

Figure 3 shows examples of the distributions on which the statistical analysis summarised in Table 1 has been performed.

FIGURE 3. Single feature performance measurements



- *Data Acquisition performance.* We have measured the performance of single Real-Time features when used in a realistic application:
  - interrupt latency and task dispatching
  - allocation and deallocation (management) of buffers, combining semaphore handling, context switching and UNIX signals
  - emulation of asynchronous behaviour
  - total event acquisition time.

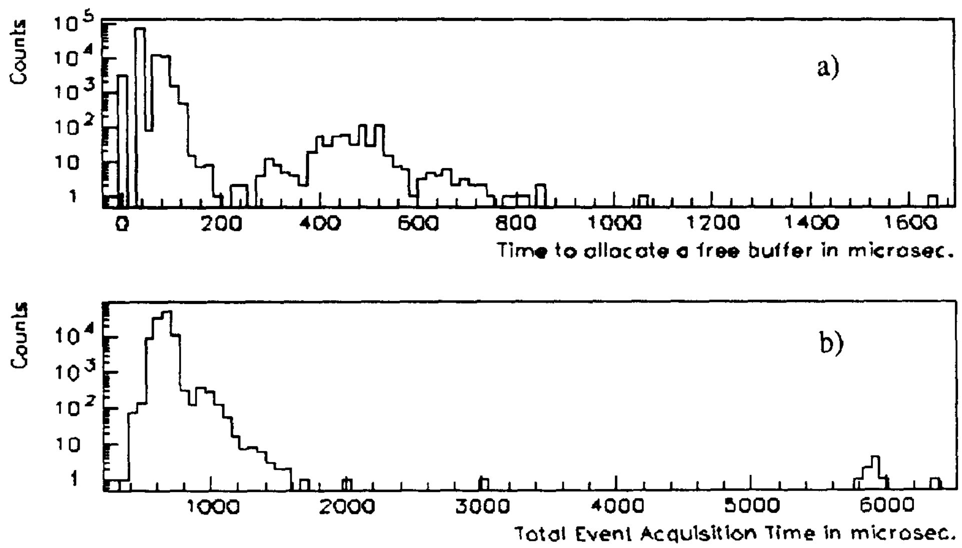
Table 2: Data acquisition performance[a]

	mean	rms
Interrupt latency	27	2
Task dispatching	47	4
Buffer management	50	12
Asynchronicity emulation	238	5
Total time	649	24

a. Default time units are µs, unless otherwise stated. All value have been corrected to take into account the overhead of the measuring procedure (i.e. 2.5 µs of Camac access).

Figure 4 is the analogous of figure 3 for the data acquisition performance measurements.

FIGURE 4. DAQ performance measurements



4.3 Conclusions on front-end environment

To the extent of the investigation performed so far, the use of Real-Time UNIX combined with RISC technology for DAQ has given positive results.

The current version of TC/IX has an excellent compatibility with non Real-Time UNIXes and provides a satisfactory and efficient Real-Time environment. The multi-processor facility and POSIX compliance are expected during the course of the year. Should the quality of the new releases be equivalent to the current one, we would consider TC/IX a good solution for the front-end operating system.

We wish to stress that the evaluation of TC/IX on the RAID 8235 card has been done in close collaboration with the two industrial partners: CDC for TC/IX and CES for the RAID card. RD13 is a beta-site for TC/IX, hence we have direct access to the engineers developing TC/IX. This has proven to be very fruitful: we have been able to get both the necessary technical assistance, in terms of bug fixes and consultancy, and to feed back our requirements into the TC/IX development process.

5. Control of the RD13 DAQ system

5.1 The run control facility

We have used ISIS to implement a run-control system as part of the data acquisition system. We run ISIS on a network of UNIX workstations and CES RAID embedded computer systems that run the TC/IX operating system.

The components of the data acquisition system, such as the read-out module and data formatter, are modelled as finite state automata, controlled by a run-control program, which sends commands to the components in order to change their states. Starting from this model, we defined a set of process groups and message formats to support the manipulation of finite state automata. A library of routines [TN 5] was implemented to support the model and provide a framework in which component-specific code could be inserted. The library is linked with the data acquisition components causing them to join the predefined process groups and establish tasks that handle the messages. The library also provides a programming interface by which the run-control program can send commands to components and be informed of the result and their current state.

The run-control program is itself modelled as a finite state machine. To interact with the run-control program, we have developed a Motif based graphical user interface from which the user can send commands and interrogate any process using the library. The Motif code for the graphical user interface of this application was generated by the XDesigner tool.

5.2 Error message facility

We have also used ISIS as the basis for an error message reporting facility [TN 6]. Data acquisition components report error conditions by calling a routine in the run-control library that broadcasts the condition as a message to a dedicated process group. The members of the process group are processes that wish to capture error reports. A filter mechanism, based on UNIX regular expressions, allows members of the group to capture subsets of all the errors reported. The graphical user interface to the run-control is also a member of this process group and displays all reported error message to a window on the screen. Another process writes error reports with a time stamp and the identification of the sender to a log file. Filters can be down-loaded to data acquisition components in order to suppress error reporting at the source.

6. User interfaces and documentation

6.1 User interfaces

Modern data acquisitions are large and complex distributed systems that require sophisticated graphical user interfaces (GUIs) to monitor and control them. An important choice to be made when designing graphical user interfaces is that of the GUI toolkit to be used. GUI toolkits control output to the screen, accept input from the mouse and keyboard and provide a programming interface for programmers who wish to built user-interfaces for their applications. Traditionally, programmers have coded the user-interfaces by hand but recently a new type of tool, called a GUI builder, has appeared on the market which allows the programmer to interactively develop user interfaces.

There are many GUI toolkits and GUI builders available and so we conducted an evaluation of some of the most promising candidates, considering only the ones based on the X11 protocol [8], the defacto standard for distributed window systems. This choice was driven by a need to run client programs on remote machines and display the windows on a local workstation. We evaluated a number of graphical interface toolkits including Motif [9], OPEN LOOK [10], and InterViews [11]. We chose Motif mainly because it has become the most widely supported GUI toolkit available and its development is driven by the membership of OSF which, to our thinking, means it is more likely to develop in a vendor-neutral manner than its competitors.

For the choice of a GUI builder, we considered VUIT [12], X-Designer [13], XFaceMaker [14] and UIM/X [15]. We chose the X-Designer graphical user interface as our GUI builder. It generates C code or UIL that can be ported to many platforms, including VMS, and requires no run-time library. It offers a specialised layout editor for the form widget and font and colour selectors, can produce an X resource file to be distributed with the final application and allows customisation of the generated code so that application specific-code can be inserted whenever each widget is created or managed.

6.2 Documentation

We considered it important to have a well organised documentation system for the project. The documentation system should be supported by software tools available on many platforms and be capable of managing a large set of documents, allowing them to be viewed on-line and to be printed. The system is intended to be used as a means of writing technical documents describing the work of the project, as a general authoring tool for published papers and, at the same time, to provide a source of on-line help to the operators of the DAQ.

Since it is often simpler to provide on-line information to the user as a combination of text and diagrams we wanted easy integration of text and graphics and the ability to navigate between documents.

Hypertext systems [16] offer the ability to cross-reference documents and navigate such references. To satisfy our needs, we surveyed a number of freely available and commercial products to evaluate their suitability, including Digital's Documentation System[12], WorldWideWeb [17], Publisher[18], DynaText [19t], X.deskhelp and FrameMaker [20].

We chose FrameMaker for a number of reasons. It is available on many platforms including MacIntosh, NexT, VAX/VMS and most UNIX machines. Documents can be transferred between different architectures by using a special FrameMaker Interchange Format (MIF). It offers excellent support for input from other popular documentation systems and comprehensive graphics, table and equation features. The hypertext capabilities allow us to use it as the on-line help facility to be linked with the graphical user interfaces built for the DAQ applications. Since it is based on X-Windows, documents can be viewed on most workstations that are equipped with bitmap screens. The next release of FrameMaker will produce documents in SGML/CALS [21] format, allowing integration with the WWW documentation project. FrameMaker's future support for SGML will allow the migration of our existing documentation to any future market leader.

7. Software engineering

As mentioned in the motivations of the project, we have tried to learn some of the principles of software engineering in order to produce software, which is more reliable and easier to maintain.

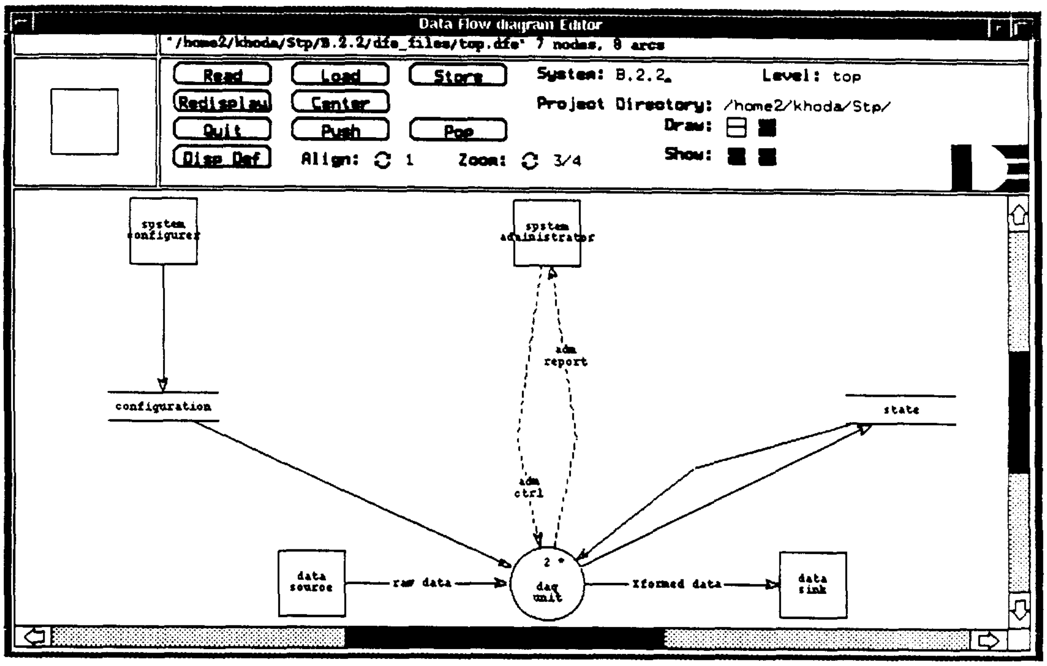
7.1 Design of the data flow protocol

We are applying the principles of software engineering to the analysis, design and implementation of the RD13 data flow protocol as described in section 3.4. It is necessary to select a software design methodology with which we can model the problem and to find a CASE (Computer Aided Software Engineering) tool capable of supporting such a methodology. The goal of the software engineering exercise is to go from the list of requirements to the executable code [TN 25].

The CASE tool we have chosen is called StP (Software Through Pictures) by IDE (Interactive Development Environments). StP is a set of CASE tools supporting analysis and design methods for software production. It offers diagram editor tools for data flows, data structures, structure charts, entity-relationships, control specifications, control flows and state transitions. There is also a picture editor and object notation editor which allows the user to associate a set of properties and values with any object in any editor diagram. A document preparation system allows designs to be printed in many formats including postscript, Interleaf and FrameMaker. All the diagram editor tools use a data dictionary which is built on an internal database. StP can be extended by writing other modules which access the information in the database. SQL code to create database tables can be generated using the entity-relationship editor. C code for the definition of data structures can be generated from the data structure editor.

To model the protocol we have used a Real-Time structured analysis method, as described in Ward/Mellor [22] and Hatley/Pirbhai [23]. Both variations of the method are supported by StP. First, the requirement model is defined, specifying what the system must do in terms of control and data flow diagrams. As an example we show in figure 5 the top level, or context, diagram which defines how a DAQ-Unit fits into the overall data acquisition system.

FIGURE 5. Protocol context diagram



Solid lines are data flows and dashed lines are control flows. Square boxes are externals and represent elements which are out of the scope of the model, but interact with it. The four externals are:

- *data source* to send data to a DAQ unit
- *data sink* to receive data from a DAQ unit
- *system manager* to exchange control signals with a DAQ-Unit.
- *system configurer* to choose the initial setup (architecture) of the system, put it in a data store called "configuration" and make it available to the DAQ-Units of the system.

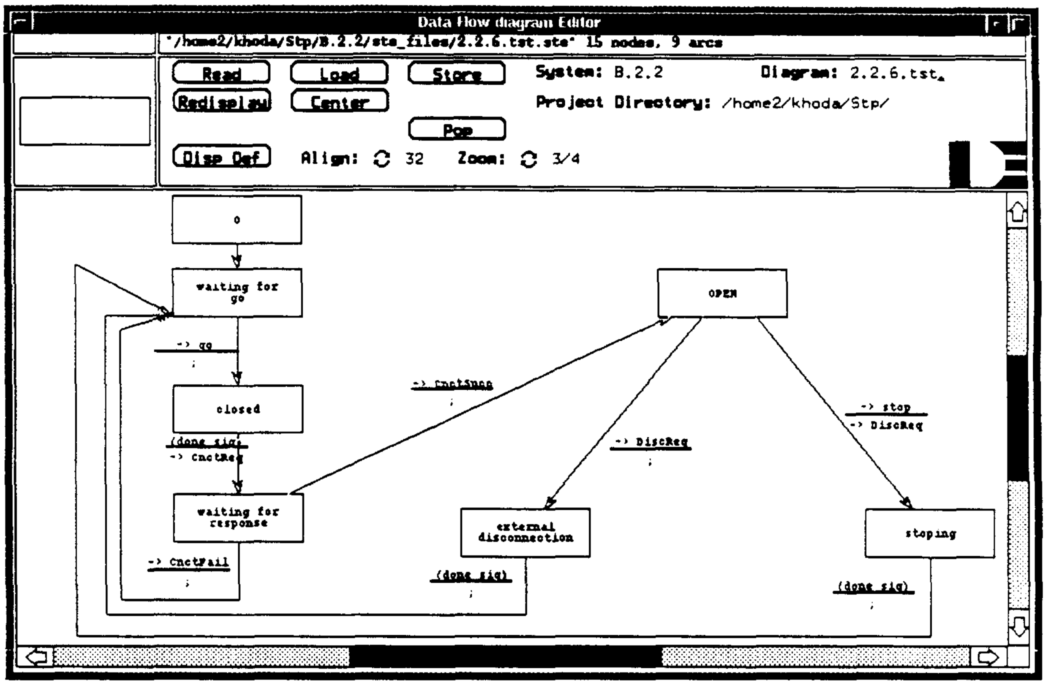
The configuration data base, as well as the other data structures used to manage the protocol, is defined according to the entity-relation data model.

The protocol itself is specified in terms of state transition diagrams. It is modeled as consisting of two parts, a server side and a client side, which are considered as separate processes. Figure 6 illustrates the client side of the protocol, a similar diagram exists for the server part. Boxes represent states, arrows state transitions, horizontal bars the external event causing the transition (above the bar) and the action (below the bar) to be undertaken before completing the state transition.

These different modeling tasks are performed by means of the corresponding StP editors, the results are then stored in the StP data dictionary and are available both to the StP editors and to user defined modules (e.g. for code generation).

Comparing what a CASE tool for protocol design should provide and what is available in StP, we find that structure decomposition is a fundamental feature that helps a step by step refinement of the system from the context level down to the specification level. This feature is supported by StP. Rigorous consistency checking between the decomposed levels of the design, as supported by StP, greatly helps to identify errors. Since a protocol is basically a state machine, the decomposition of a state machine would make the definition clearer. Unfortunately StP does not support this feature.

FIGURE 6. The client protocol



The schema drawn with a CASE tool is an important aid when discussing details of the design. A customizable paper document preparation feature is essential to allow the designers to discuss the problem away from the workstation screen.

A very useful testing feature is a state transition simulator. Such a tool, though not offered by Stp, is important for early protocol testing. It should be able to run several interacting state machines concurrently and allow the user to intervene to test different situations, send control signals and set break points.

Although normally one would leave the CASE tool at the completion of the design phase and proceed to implement the required system by hand, it is possible to use much of the information contained in the diagrams of the design to generate a portion of the required code automatically. While StP does not provide a code generator as such, because of its open architecture, it is possible to retrieve the details of the design from the Data Dictionary and use it to generate code. This implies extra effort from the developer, which should not be the case when using a software engineering tool like StP.

In short, we can say that, StP has many interesting and important features but the lack of a state transition decomposition and simulator makes it weak for protocol design.

8. Status of the DAQ development

The first task that we have defined as a preamble to any DAQ development was the thorough investigation of all the components of the system, according to the design outlined in the proposal and further specified along the lines described in section 3. This phase is now complete and the elements of the system are now individually mastered to the necessary extent (sect. 8.1). The first integration of the system elements is in a DAQ prototype (sect. 8.2), whose main role is the validity check of all the elements and the measurements of performances. The status of the use of commercial products and of the available RD13 computing resources are summarised in sections 8.3 and 8.4 respectively.

8.1 Basic function libraries

The completion of the investigation phase consisted in the development and the integration by the RD13 team of the following function libraries [TN 7-12-13-24], which constitute now the basis for the construction of DAQ modules:

- VME Library: VME address space mapping and Interrupt handling
- VIC Library: VIC 8250 management
- SVIC Library: SVIC 7213A management
- RAID Library: management of the internal RAID resources
- Camac Library: implementation of the full ESONE specification.

The following Table summarises the availability of the libraries on the different RD13 platforms, the RAID running both a debug monitor from IDT (Integrated Device Technology) and TC/IX, and the SUN interfaced to VME via the SVIC 7213.

Table 3: Library implementation

Library	RAID IDT	RAID TC/IX	SUN
VME	yes	yes	yes
VIC	yes	yes	yes
SVIC	n/a[a]	n/a	yes
RAID	yes	u/d[b]	n/a
Camac	yes	yes	yes

a. not applicable
b. under development

8.2 DAQ prototype

The development schedule proposed by the RD13 collaboration foresees the construction of a minimal system, a simplified implementation of the RD13 DAQ requirements, as a first step towards the full data acquisition environment. While it embodies the concepts of scalability and modularity, i.e. it follows the RD13 DAQ architecture, the minimal system contains embryonic implementations of the required functionality.

Prior to the development of the minimal system, a first DAQ prototype, which integrates all the system components, has been developed with the double aim of fully debugging the basic function libraries and of evaluating the validity of the element choices, once integrated in a fully functional DAQ system, by measuring the Real-Time performance of the DAQ structure. The prototype system is an initial, laboratory version of the minimal system. Its scope is limited to that of a test bed for the ideas developed in RD13 and their integration into a running system. As such it is not intended to run in any production environment. In other words, the DAQ prototype is a data acquisition where attention has been given to integrating all the data acquisition functionality in an operational setup, but no attention has been put into the operability of the setup.

In this section, we describe the DAQ prototype that we have implemented and give an account of the measurements of the system performances that we have made before proceeding to the full exploitation of the proposed architecture.

8.2.1 DAQ prototype requirements

The prototype system has been constructed to satisfy the following criteria:

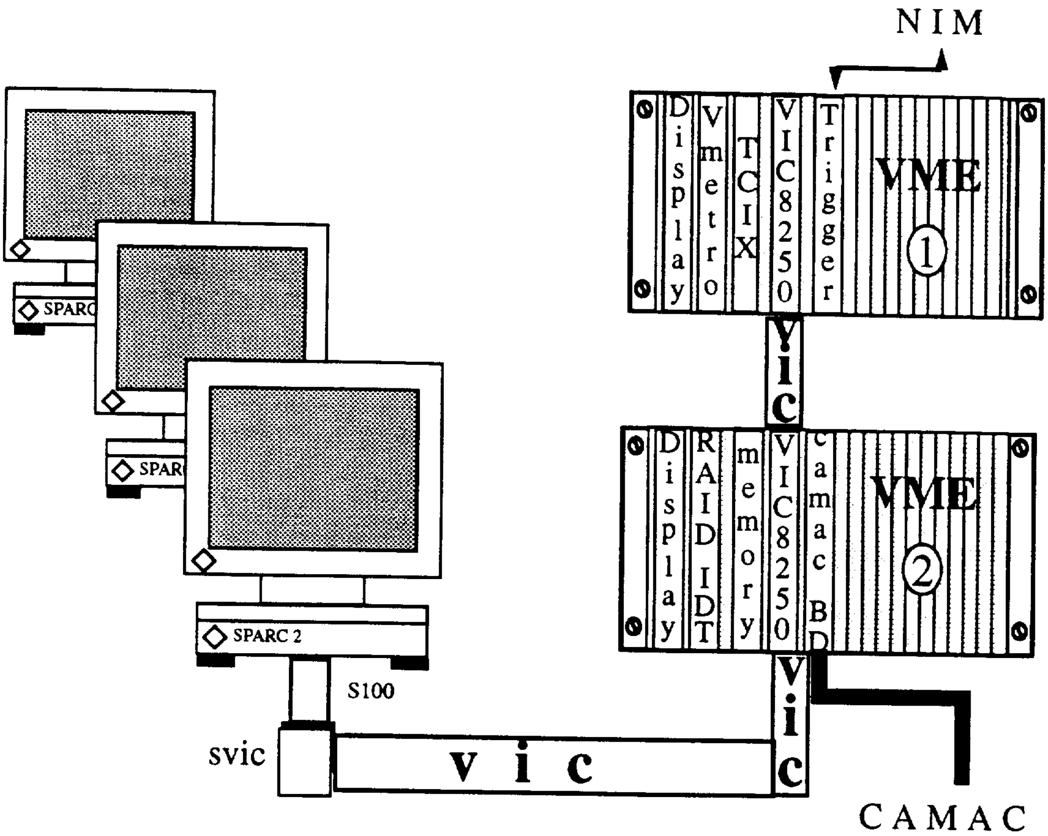
- take data from a selection of electronics modules, sitting both in Camac and in VME, in response to a trigger (see section 8.2.2)
- read and manipulate the data in a DAQ-unit whose main dataflow chain will run on a single RAID-TC/IX processor
- log the events either on disk or tape, via the RAID
- provide control and user interface on a SUN connected via SVIC/VIC to the VME hardware
- provide monitoring of events on the SUN, the RAID and the SUN belonging to the DAQ-unit concerned
- provide a data access library for the data flow, the control and the system configuration data bases.

8.2.2 Hardware configuration

The proto-DAQ runs on a hardware configuration which reflects the basic RD13 hardware DAQ architecture (figure 5), namely:

- CBD module, connected to a Camac crate, and a VME memory module (CAMAC modules and VME memory will provide "front-end" electronics) in one VME crate
- RAID-8235 processor, running TC/IX, reading the event from hardware and logging the data on a medium (disk or tape), in a second VME crate.
- VIC 8250 module in each VME crate
- SUN workstation, linked to the VIC bus via a SVIC-7213 module
- SUN workstation cluster providing development, user interface and controls
- VME trigger module [24 and TN 23] to generate VME interrupts as a result of external triggers.
- VMETRO, a VMEbus analyser.

FIGURE 7. Hardware configuration for DAQ prototype



8.2.3 Software configuration

The following software components are part of the proto-DAQ:

- Data acquisition modules for event read-out, formatting, recording and monitoring (on the SUN and on the RAID)
- Data flow protocol, to move data between data acquisition modules, consisting of a single DAQ-Unit which includes the RAID and the SUN
- basic run-control system capable of configuring the DAQ, starting and stopping the data taking from a graphical user interface
- initial implementation of the Error Message Facility
- first implementation of the data base framework, based on StP for data modelling and code generation, and ORACLE as the back-end DBMS
- System Configurer, a data base of the system configuration, its Data Access Library and a program capable of initialising all the hardware components
- Run-Time Libraries, providing the basic services common to many modules

The basic concept of the RD13 DAQ architecture, the DAQ-Unit, is implemented in the DAQ prototype to span a RAID and a SUN, the shared memory being provided by the RAID slave memory. Shared memory primitives for both local (in the RAID) and remote (from the SUN via the SVIC/VIC/

VME) access had to be implemented as well as primitives for synchronisation [TN 17]. Distributed computing features, in particular the Run Control and the Error Message Facility, exploit fully the capabilities of ISIS.

8.2.4 Measurement of performances

The performance of the proto-DAQ in terms of number of events acquired per second are summarised in Table 4. One has to stress that the measurements were done on zero-length events, i.e. they correspond to the full overhead of the acquisition process. The three figures relate to the following conditions:

- Single task: this is a configuration of the DAQ which consists of the read-out module only; it is typical of data taking without tape recording (e.g. calibration).
- Two tasks: this is the classical configuration, with read-out and tape recording modules.
- Four tasks: this represents a data flow chain, when a formatter and a filtering module are included.

Table 4: DAQ performance

	Hz
Single task	2450
Two tasks	1300
Four tasks	580

8.3 Use of commercial products

In our proposal [1] we have clearly stated our conviction that in many aspects of the development of a sophisticated DAQ system one could take advantage of the enormous industrial investment of manpower and resources in software production. We have selected commercial products for the implementation of various aspects of the system where we judged their use advantageous and more economical.

We want to emphasise the importance of developing a scheme for the efficient use of commercial products: in the challenging environment of a full size LHC experiment, the resulting reliability and maintainability will be of enormous benefit.

Many of the tools selected by RD13 have proved to be the market leaders in their own field. The success of our tool evaluations is reflected by the fact that many of them are now capable of interoperating. For example, the StP CASE tool now produces printouts of diagrams in FrameMaker format. The next releases of both X-Designer and StP are integrated with the Saber C++ programming environment. This programming environment provides the application programmer with many of the tools required for program development including a symbolic, graphical debugger, integrated language interpreter and cross-reference mechanisms.

Some of the outcomes of the RD13 evaluation have been adopted by other groups at CERN, namely X-Designer and ISIS. Furthermore, there is an increasing interest in the RAID-TC/IX system.

8.4 Computing resources

To provide a development environment the RD13 laboratory has been equipped with SUN workstations clustered around a SPARC2 used as a server [TN 20]. Clusters of RISC workstations provide large amounts of CPU power which is not always fully exploited during the development phase. This is also the case of the RD13 SUN cluster. We have, therefore, installed the NQS batch queue manager, extending it to include the front-end Real-Time UNIX processors [TN 19]. We have tested the performance of running CPU intensive applications, such as GEANT based Montecarlos, and we have found satisfactory results without compromising the main activity of the RD13 system development. We are, therefore, encouraging the use of the available CPU power for LHC detector simulation studies.

Furthermore, having promoted the use of a number of commercial products, we have been acting as the central serving place for some of them (Saber C++, StP, XDesigner) and we are hosting users from other projects and experiments.

9. Proposal for the next phase

Given the positive results that we have obtained from the investigations and evaluations performed in RD13 Phase 1, we propose to continue the project following the lines already indicated in the proposal. We see five main directions for Phase 2:

1. completion and exploitation of the minimal DAQ system in a test beam setup
2. R&D activities in the laboratory development system
3. software engineering
4. exploitation of the system for event building studies
5. initiate contact with industry for possible collaboration on software development.

9.1 Test beam system

Some consolidation is necessary to evolve the DAQ prototype developed so far (section 8.2) to a minimal functional implementation of the RD13 DAQ system as described in section 3. We maintain our original plan to operate a data taking system for a detector R&D at a test beam right from the availability of the minimal implementation.

In conjunction with RD2, we have obtained a beam period on the H2 line at the end of the summer, where we plan to readout silicon counters with pipelined electronics clocked at LHC speed (67 MHz). The initial DAQ architecture is the one outlined in figure 7, although details of the integration are presently under discussion.

As indicated in the proposal, we plan to continue the exploitation of such a test beam system for the development of DAQ functionality in conjunction with that of detector R&D's readout electronics. Should the features of the RD13 DAQ respond fully to our expectations, its architecture makes it very appropriate for the next phase (1993) data taking of certain detector developments. One should then consider the possibility of organising a 'portable-DAQ' based on the RD13 structure and using the most mature components. Further investigations are necessary to assess the interest of such a tool and to establish the resource requirements.

9.2 Development system

There are several directions in which we propose to continue our R&D activities:

1. improve the functionality of the system components
2. extend the DAQ-Unit to a multi-processor environment
3. evaluate the Object Oriented Data Base technology for online applications.

Most of the directions follow naturally from the choices made in the first phase, since much attention has been given to the availability of development paths in every product selected.

9.2.1 Functionality improvements

An example of hardware improvements is in the area of inter-crate connection. The latest implementation of the VME Vertical Inter-Crate connection has a much bigger on board memory (up to 16 Mbytes) and supports hardware data broadcasting (reflective memory). Although it was foreseen to use this version of the VIC module (VIC-8251) already in the first implementation of our system, we have postponed its use due to the non-availability of the Sun/VIC-8251 interface (SVIC-7213B).

Improvement paths have already been defined also in software elements, such as the ones based on ISIS and related products, on a new available release of XDesigner for the user interface, in addition to the upgrading of the features of the basic DAQ modules.

- *Run-control.* Built on top of ISIS is a facility, called Meta [25], for controlling distributed applications via the use of guarded commands. The run-control program could then be written in terms of such commands instead of using ISIS directly. This offers the advantage of simplifying the implementation of the run-control library by allowing the programmer to think more in terms of the states of the data acquisition components rather than ISIS messages and process groups. It also means the run-control program can be seen as a set of rules, written to handle particular situations, which are always ready to fire.
- *Future of graphical user interface.* We have used XDesigner to develop graphical user interfaces to our database, system administration software and run-control. For the run-control, we would like to improve the interface so that it is icon based and permits direct manipulation. For example, we would like to have the components of the DAQ system, such as the data formatter and read-out modules, appear as icons on the screen. A data driven graphical editor tool based on the ideas developed in the InterViews research project, called DataViews, is available and can be used in conjunction with XDesigner.

9.2.2 Multi-processing

This consists in extending the DAQ-Unit to span more than one front-end processor. We plan to pursue two main directions of work: the evaluation, and further the exploitation, of the multiprocessor model provided by TC/IX and the management of resources in a distributed system. In the former case we aim to redistribute the DAQ-Unit modules over several processors, TC/IX provides the necessary features to make this redistribution transparent to the applications; these features have to be evaluated with a particular emphasis on the performance aspects and the limits of their applicability. In the latter case, commercial products such as ISIS, combined with a system description data base, may provide the necessary functionality to implement a protocol for coordinated and safe access to system resources (e.g. to coordinate the access to a hardware module).

9.2.3 Object oriented data base framework

We intend to evaluate the applicability to an online environment of Object Oriented Data Base technology, by re-implementing the Data Base framework of section 3.5 using a commercial OODBMS. One such OODBMS is ONTOS [26] which provides all the required facilities for Real-Time use (eg. an in-memory cache) and implements a data model which matches a programming language (C++) thus making easy and efficient the implementation of data access routines. ONTOS also includes graphical tools to interactively design the data model and to browse the data base contents. The outcome of the evaluation will be twofold: we will gain experience with this new technology and we will be able to prove (or disprove) the expectations we have concerning the Real-Time use of such commercial DBMSs. Another OODBMS is O2 [27] which offers similar facilities. We will decide in the near future which of these databases we will use to perform the evaluation.

9.3 Software engineering

The StP exercise to design the data flow protocol (section 7.1) for the DAQ system proved to be a good introduction to software engineering and has allowed us to gain experience in this domain. The design is now complete and we are considering various means to access the CASE tool's data dictionary in order to automatically generate the code to support the protocol. On completion of this exercise, we plan to redesign the protocol using another CASE tool, such as ARTIFEX [28], in order to make a comparison of the CASE tools and explore other aspects of software engineering.

9.4 Event building

In the first phase of the project, event building studies have constituted a somewhat independent development. The ideas was that, while developing the RD13 DAQ, simulation and bench testing of the proposed event builder techniques were done in parallel as preparatory work before the integration. We report here the status and present the program of the studies in the three areas that we had proposed: Simulation, System prototyping of a optical fibre star network and Integration of a HiPPI based implementation.

9.4.1 Simulation

Formal Description Techniques (FDTs) are used to describe the behaviour of systems. Formal Specification Languages, such as Estelle, LOTOS and SDL, are proposed to ensure the correct specification and implementation of communication protocols. Intended primarily for the description of 'logical' behaviour of systems, the formal specifications do not address performance aspects, such as the maximum throughput of a link, end-to-end transmission delay and service degradation from error occurrence. Traditional approaches consist of creating performance models with a different formalism to deal with performance aspects (e.g. Verilog, MODSIM), losing in this way the 'logical' properties of the system. We are studying a method which allows the combination of the logical correctness of specification with the evaluation of its performance. in particular, we are using the newly-proposed Timed-Lotos, redefining the treatment of time, and extending it in the near future with the concepts related to the performance models of one of the actual simulation languages [29].

The simulation activities have also been directed to a more specific modelling of event building on two different subjects:

- a model to use a FDDI switch (section 9.4.2) in a DAQ system is now fully developed in Verilog,
- a model based on the HiPPI protocol for data transfers and commercial HiPPI switches for data merging and distribution has recently been started using MODSIM.

We have chosen these two systems because of the possibility of testing them on existing hardware, as will be described in the following sections.

As a continuation of the simulation work, and taking advantage of the experience gained, we plan to extend the model to other aspects of a DAQ system for LHC, to eventually reach a full system simulation. We see this work as based on RD13 resources, but in collaboration with other people and institutions.

9.4.2 Optical fibre star network

Financial support for the GaAs transceiver gate-array for optical communication has been agreed by the INFN only in February 1992, after many iterative discussions with industrial partners, because of the very high cost of development of the chip. The main features of the chip are listed here:

- 32 or 40 bit transmission (auto-correction of single bit error can be implemented with the 40-bit).
- Manchester coded. Simulated in laboratory with GaAs components up to 800 Mbit/s, this code seems to us a good solution for the final gate-array because of the high simplification of the clock recovery system and the natural balancing between 'ones' and 'zeros'.
- The present project does not have the laser driver, which will be included in the multi-chip system to be defined as next step. The advantage of the laser driver is the reduced power up and down times, making its use suitable for different network topologies.

Switch components are also under study for the use of intelligent high performance switches that are being realised for applications in high speed switched networks (full duplex FDDI, HiPPI). Tests of a FDDI switch connected to RISC machines are planned for the summer. Use of the switch internal buffering and the programmability of the switch control functions will be tested for critical parameters, such as host interface performance, high level protocol characteristics and latency times.

9.4.3 HiPPI based implementation

A HiPPI-VME interface based on the CES RIO 8260 module has been developed in the ECP/EDU group [30] and will be available in prototype form in June this year. The RD13 proposal referred to this card as a possible candidate for event building studies. The high level of compatibility between this HiPPI-VME module and the DAQ processor of RD13 (both are based on the MIPS R3000 processor) make the integration of such an event builder prototype in the RD13 DAQ system particularly convenient.

We propose to install a minimal setup, based on the layout of fig. 4 of Addendum 2 of the RD13 proposal, with two sources, two destinations and a parallel switch, and to proceed in parallel with the simulation of event building algorithms and their implementation in the hardware setup. Although the performance of these components is not yet suitable for the expected bandwidth of LHC experiments, we consider such investigation as a good reference. Furthermore, we see a clear upgrade path for the system proposed: CES is planning the development of a two-processor RAID card, based on a MIPS R3000 running TC/IX and the new 64-bit MIPS R4000. Bandwidths of the order of 40Mbytes/s on the VME bus and of 100 Mbytes/s on the HiPPI I/O are expected.

The integration of the HiPPI-VME interface in the RD13 setup has an interest of its own as DAQ supporting input channels of tens of Mbyte/s, as needed, for example, by RD6 for the 1993 test beam setup.

9.5 Software developments with industry

The question of whether it is more economical to proceed fully with in-house development rather than sub-contracting parts of complex software systems to selected industrial partners is becoming increasingly important. The difficulties generated for the long term maintenance of software developed in our field, where the turnaround of people can be very fast, should be carefully compared to the cost of industrially developed and maintained software.

At the time when contacts with industry are being established at the institution level, we believe that RD13 is an ideal ground for one or more pilot projects. Commissioning software developments to industry in an effective manner implies following certain procedures, starting from the preparation of a clear requirement document and ending with the capability of performing software reviews and testing. In other words, there is a level of professionalism involved, to which our environment is not normally exposed. In view of that, we have 'played the exercise' with the development of the RD13 Data Flow Protocol (see section 7), for which the software engineering expert, member of the ECP/PT group, was not a member of the RD13 DAQ team and the communication was kept at a certain level of formality with the preparation of a requirement paper [TN 4] and regular reviews.

10. Budget and resources

The investment required to develop the full program outlined in section 9 is summarised in table 5.

Table 5: Material budget

	Global cost (kSF)
1. RD13 development lab.	130
2. Event building equipment	35
3. Test beam	115
4. Software	70

Comments on the budget request:

- CERN is expected to contribute about 1/3 of items 2 and 3, and about 1/2 of items 1 and 4, for a total of 150 kSF. This budget does not include the financing of a SPARC Server and TC/IX licences, which had been assigned on the previous budget, but has not yet been spent. We expect to need the corresponding funds this year and have it added to the present budget. It amounts to a total of 70 kSF.
- Also not included are provisions for maintenance, CERN stores and electronics pool for a total of 30 kSF. This implies the granting of a CERN Electronics Contribution (CEC) from the pool of 200 kSF at the rental fee of 4%. Such equipment is for both the development laboratory and the test beam setup.
- Provisions for travel and training for specific RD13 activities (collaboration meetings, visits to industries, specialised courses, RD13 presentations to conferences) are normally not made in the standard group exploitation budgets. Given the 'heterogeneous' composition of the RD13 CERN team, with members coming from four different PPE and ECP groups, we insist on the opportunity of having the necessary fund, which we estimate at 40 kSF, assigned directly to the RD13 budget rather than via the division groups.
- Item 2 does not include the purchase of a commercial HiPPI based switch, which, being an expensive item, we foresee to loan for limited periods.
- Item 3 does not include detector specific electronics and equipment, which we expect to be responsibility of the detector R&D collaboration.
- Item 4 only includes the payment of RD13 front and back-end licences and some provision for specific software evaluation (e.g. ONTOS, Artifex,...). We expect that the funding of expensive site-wide licences and the financing of pilot projects for industrial collaboration are done via a special contribution for software investment in ECP, as was the case last year.

Acknowledgments

We gratefully acknowledge the active support of the ECP/PT group to the software engineering developments of our project. In particular, we appreciate the valuable contributions of P. Palazzi and J. Harvey to the organisation and review sessions of the data flow protocol design. We also deeply thank D. Klein (ECP/SA) for her precious work in the organisation and maintenance of the RD13 documentation system.

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] K.P.Birman et al., The ISIS SYSTEM MANUAL, Version 2.1.
[3] R.Bauer, A review of LynxOS Unix Review september 1990.
[4] Control Data, TC/IX Users's Guide, CDC May 1991.
[5] R.Russel, G.Mornacchi, VOS a virtual operating system, CERN July 1990.
[6] B.A.Kingsbury, The Network queueing system, Sterling Software.
[7] Real-Time extensions for portable operating systems. P1003.4/D11.IEEE.
[8] R.Scheifler J.Gettys. The X window system. ACM Transactions on Graphics, 5(2):79--109,April 1986.
[9] Open Software Foundation. OSF/Motif Programmer's Guide, 1.1 edition.
[10] Sun Microsystems, Inc. OPEN LOOK Graphical User Interface Functional Specification, 1990. Version 2.0.
[11] M.Linton P.Calder J.Vlissides. Inter Views Reference Manual. Stanford University, 1991. Version 3.0.

[12] DECWindows Digital Equipment Corp.VMS DECWindows Motif Guide to ApplicationProgramming,August 1991.
[13] Imperial Software Technology. X-Designer User Manual.
[14] Non Standard Logics. XFaceMaker 2 User Manual. Version 1.1.
[15] Bluestone Consulting Inc. Getting Started With UIM/X Professional.
[16] B.Shneideman G.Kearsley. Hypertext hands-on!: an introduction to a new way of organizing and accessing information. Addison-Wesley, 1989.
[17] T.Berners-Lee et al. World Wide Web; An Architecture for Wide-Area Hypertext. CERN, 1991.
[18] ArborText, Inc. The Publisher User Guide.
[19] Electronic Book Technologies, Inc. DYNAText System.
[20] FRAME MAKER, International Version 3.0, FRAME Technology 1991.
[21] International Organization for Standardization. Information processing -- Text and office systems -- Standard generalized Markup Language (SGML). ISO 8879 1986 (E).
[22] Ward & Mellor, Structured development for Real-Time systems, Prentice Hall, 1986.
[23] Hatley & Pirbhai, Strategies for Real-Time system specification, Dorset House, 1987.
[24] Ph. Farthouat, VME Trigger Module, ECP/EDU 2.11.1990.
[25] M.D.Wood, The Meta Toolkit Version 2.1 Functional Description.
[26] ONTOS Reference Manual release 2.1, ONTOS Inc.1991.
[27] O2 Technology, The O2 System.
[28] ARTIFEX, ARTIS 1991, Artifex Environment User Guide.
[29] G.V. Bochman and J.Vaucher, Adding Performance Aspects to Specification Language, in Protocol Specification, Testing and Verification VIII, p. 19-31, eds. S. Aggarwal and K. Sabnani, Elsevier Science Pub. B.V. (North Holland), IFIP 1988.
[30] E.Van der Bij, R10 8260 option 8260/4, HiPPI source daughter module users's guide, CERN/ ECP, 4.11.1991; CES Data Sheet, Intelligent I/O Boards, Creative Electronic Systems, Geneva.

List of RD13 Technical Notes

[TN 1] L.Mapelli, RD13 Workplan - Phasel, 1.11.1991
[TN 2] D.Klein, How to produce a RD13 Note, 4.12.1991
[TN 3] G.Mornacchi, RD13 Dataflow, 20.12.1991
[TN 4] G.Mornacchi, RD13 Dataflow Requirements, 22.1.1992
[TN 5] R.Jones, Run Control for RD13, 25.11.1991
[TN 6] R.Jones, Error message Facility for RD13, 27.11.1991
[TN 7] S.Buono, Basic Libraries for IDT Monitor, 13.1.1992
[TN 8] R.Ferrari, Some basic informations on interrupt handling within IDT, 20.12.1991
[TN 9] R.Ferrari, VME Interrupt Requests from CBD 8210, 20.12.1991
[TN 10] R.Jones et al., Using Motif in RD13, 26.11.1991
[TN 11] G.Mornacchi, RD13 Database Frame Work, 26.11.1991
[TN 12] G.Ambrosini, Basic Libraries for SVIC/VIC Interface, 20.1.1992
[TN 13] G.Fumagalli, C.Rondot, Basic Libraries for TC/IX Environment, 3.2.1992
[TN 14] RD13 Team, RD13 Prototype Minimal DAQ, 17.2.1992
[TN 15] G.Mornacchi, Data Flow Protocol Prototype, 23.1.1992
[TN 16] M.Huet, Event and Data recording formats, 22.1.1992
[TN 17] P.Y.Duval, Processes Synchronisation in a VIC/VME System, 29.1.1992
[TN 18] G.Mornacchi, F.Tamburelli, Directory Organisation for Production Software, 17.2.1992
[TN 19] G.Mornacchi, F.Tamburelli, Batch processing in the RD13 Computer System, 29.1.1992
[TN 20] F.Tamburelli, RD13 Cluster Management Utilities, 29.1.1992
[TN 21] G.Mornacchi, Real-Time requirements, 8.3.1991
[TN 22] R. Jones, L.Mapelli, G. Mornacchi, Real-Time UNIX (TC/IX) evaluation, 10.3.1992
[TN 23] C.Rondot, Using the VME trigger module through macros, 12.2.1992
[TN 24] G.Ambrosini, SVIC 7213/ VIC 8250, 30.1.1992
[TN 25] A.Khodabandeh, Design of a data flow protocol with StP, 4.3.1992
[TN 26] R.Jones, G.Mornacchi, G.Polesello, Using ISIS and META for Run Control in RD13, 1.3.1992.