Run II Computing and Software Plan for the DØ Experiment

DØ Run II Computing Planning Board

January 31, 1997

Abstract

The DØ plan for developing its computing systems for Run II, including both hardware and software components, is described in this document. This plan ranges from operation of the DØ detector itself through the processing of its data to the production of physics results. The plan includes organization, schedule, staffing, and budget requirements to achieve a set of milestones culminating in operational systems in July 1999.

Contents

1 Overview

The DØ experiment, like all large HEP detectors, will utilize an extensive suite of customized software running on a complex array of computer systems to perform the tasks of data acquisition, event reconstruction, data storage and retrieval, and physics analysis. Limitations in computing technology and its affordability, at various stages in this process, inevitably constrain the final return from the experiment. The goal in developing the computing and software system is to alleviate these constraints as much as possible, by targeting cost effective resources and maintaining sufficient flexibility to address revisions in requirements and to exploit advances in technology. An effort will be made in this document to indicate where such constraints exist and what choices were made to address them. In addition to hardware constraints, software limitations can be equally important and an effort will be made to indicate how the technical options selected offer an appropriate compromise among the competing demands of reliability, performance, flexibility, and the practicality of meeting required deadlines for a functioning system.

1.1 The Computing Problem

The primary functions which the DØ Computing and Software systems must perform are:

1. The configuration of the data acquistion system and control of the operation of the detector.
2. The selection of interesting events by the software (Level 3) trigger.
3. The recording of events.
4. The monitoring of the detector, accelerator information, and event data.
5. The calibration of the detector.
6. The Monte Carlo simulation of physics processes and detector responses for the purpose of understanding acceptances and resolutions.
7. The reconstruction of events. This is accomplished by processing the raw data through a CPU-intensive reconstruction program, running on a large farm of processors (provided, as in Run I, by the Computing Division). The ability to have the reconstruction process approximately keep up with the acquisition of events is an important ingredient in the ability of the experiment both to monitor its data and to produce timely physics results. The size of the processor farm and the efficiency of the program itself are of paramount importance in determining throughput, but also important are the following requirements:
    (a) The program must be reasonably platform- and operating-system-independent, to allow for maximum flexibility in farm hardware purchases.
    (b) The program and associated release procedures must be designed for robust execution, effective error recovery, and ease of verification.
    (c) The program must be capable of a reasonably effective reconstruction pass at the beginning of serious data-taking, in order to avoid falling behind the real-time collection of data and to avoid the need to waste resources by reprocessing of large portions of data.

8. The selection/identification of data of interest in the output of the reconstruction; so called data *mining*. Since the volume of data emerging from the reconstruction farm will be very large, the efficiency of access to these data will determine to a considerable extent whether a particular physics analysis can be completed in a reasonable time frame. The problem is complicated by the wide variety of access requirements for different classes of physics. For example, while many physics analyses can be conducted using a modest subset of the full data available for each reconstructed event, the events used in B-physics and W mass analyses will need virtually all of the available information for the events used in those analyses. QCD analyses, on the other hand, can use an even smaller subset of the data for each individual event, but will need very large numbers of events. These diverse requirements coupled with performance and cost considerations which dictate that the data volumes read during analysis be minimized and that multiple copies of data not be stored on expensive storage media, will require innovative data organization and access strategies.

9. Physics analysis, based on *mined* data. Once the data have been identified and organized for access, there remain the systems for the actual physics analyses. There are many issues to be considered here and the final direction to take need not be completely specified as early as the run-time system decisions. Some of the points for consideration are:

    (a) Central vs. distributed systems for analysis
    (b) Size and form of non-Fermilab hardware contributions
    (c) Systems that allow effective analysis participation from remote institutions
    (d) Effective version control, documentation, and verification of (non-production) analysis software
    (e) New software tools in the tradition of PAW/MINUIT/etc.

In order to accomplish the above primary functions the DØ Computing and Software system must perform a number of corollary functions which include:

1. Software development support

2. Population of various databases with information regarding the detector, the data and the data processing, and provision of tools to access that information efficiently

3. Facilitation of internal and external communication.

1.2 Scale of Quantitative Requirements

Performance requirements for the computing system in Run II, derived from expected collider luminosity and duty factor, estimated detector and software performance, and physics goals are approximately an order of magnitude greater than those for Run I.

1. Raw data will be recorded at the rate of approximately 150 TeraBytes (TB) per year resulting in total data accumulations of up to 400 TB per year. Access patterns for these data are expected to require that $\approx$ 5 TB of it must reside on disk and $\approx$ 80 TB in robotic storage, with the remainder sufficiently rarely accessed to permit shelf tape storage. The form and organization of the premium forms of storage, as will be discussed later, substantially impact the price/performance.

2. Compute intensive production operations including reconstruction and Monte Carlo will require approximately 70,000 MIPS of installed compute power. I/O intensive and interactive computing of the the data mining and physics analysis type will require 50,000 MIPS of installed compute power. The I/O requirement and the need to effectively utilize physicists' time will lead to different optimizations for various of these CPU requirements.

3. Bandwidth requirements for compute intensive production operations are modest while those associated with data mining and physics analysis will require aggregate disk access rate capabilities of 250 MB/sec. The robotic tape storage system must be capable of supporting aggregate access rates of 25MB/sec.

4. The technical effort required to develop the computing system described in this document is one of the most critical and scarce resources. One might have expected that existence of a Run I would imply that the bulk of the software for Run II would already exist. However, of the three major detector subsystems in Run I, one will disappear completely and a second will undergo major modification. The disappearing one is replaced by a system with very different software requirements (tracking in a magnetic field versus straight-line tracking; finding secondary vertices). So, much of the DØ reconstruction system must be replaced. With the evolution of the detector into one with a central magnetic field, the basis for particle identification and detector calibration changes, implying that the analysis programs and calibration procedures will also change significantly. Externally driven changes in hardware platforms, operating systems, languages, and foundation utilities further dictated a fundamental rewrite of the DØ computing system. It is estimated that this will require 180 FTE-years of software effort through the first year of operation.

1.3 Strategic Considerations

Much has changed with regard to computing and software since the original DØ proposal was accepted in 1983. The PC revolution and the associated explosion of software technologies are recent examples of these changes. It is no longer true that High Energy Physics (HEP) is one of the few non-defense-related consumers of sophisticated computer and software technology. However, it is still true, in the words of one of the computer professionals who have recently taught C++ and Object-Oriented Design courses at Fermilab, that "Your domain is not typical." Early in 1995 DØ began to lay out its plan for transition from the established HEP modes of computing to modern mainstream techniques. Given the limited resources of a single experiment, this transition will benefit greatly from cooperation and coordination with the Computing Division and CDF, and significant progress has been made in the last year in establishing joint efforts among these three parties in several areas of Run II planning.

A review of the DØ Run I computing experience allows one to extract useful lessons which are applicable to Run II. DØ's Run I computing systems were based, at the beginning of the run, on several distributed clusters running the VMS operating system: the online cluster (DØ), the fileserver cluster (D0FS), and the analysis clusters (FNALD0/D0SFT/D0TNG). However, the production farms ran UNIX, so that all production software had to be verified and maintained on two flavors of UNIX as well as VMS. The experiment-specific offline software was written completely in Fortran (Fortran 77, augmented by the most widely-supported subset of VAX Fortran extensions), while the run-time system included programs in Pascal, Fortran, and C. Vendor-supplied compilers for each supported platform were used. The functions of memory management within programs and of data portability across platforms were supplied by the CERN product ZEBRA. The calibration databases were implemented in DBL3 (a free product, loosely supported by the L3 experiment). Production information and hardware specifications were recorded using the DEC relational database RdB. Production file management was handled by FATMEN, a CERN-supported free database product. Event display graphics programs used a commercial product DI3000, which later had to be emulated using X11 calls by a DØ collaborator owing to the non-availability of DI3000 by the end of the run; online display graphics typically used Motif calls.

Listed below in descending order of importance are problems and bottlenecks encountered with the Run I system.

1. Platform portability and program verification required a long (and ever-increasing) time for each release, eventually posing a significant deterrent to making major changes and upgrades to the reconstruction.

2. Access to the μDST's (a subset of event information which was kept on the fileserver disks for the entire data set) was limited by network hardware and software problems. Nevertheless, it remained the primary mode of data access and worked sufficiently well for the needs of most analyses.

3. Access to the full event information (stored on 8 mm tapes) was slow, difficult, and painful, delaying many analyses and limiting plans for others.

4. Since several types of database were used, most not commercially supported, support for the databases consumed a great deal of effort, and still did not totally meet the demand for increased functionality.

The basic issue associated with the first problem described above reduced to the inability to test pieces of the program independently and guarantee that they would then function together as a system. The primary reason for this was the coupling among different sections of the program introduced by the lack of any notion of modularity or data encapsulation which could be easily implemented in Fortran. The net effect of this problem was to drive upward the time required to make a working release including changes to many areas of the program at once. The appropriate fix is to use a modern programming language *and to design explicitly for separate testability of the code packages.*

The second and third problems are addressed by a redesign of the computing model to have the analysis jobs running on a central multi-processor system with direct access to a large disk set and to highly reliable robotic mass storage, reducing the need to move data across the network. Whether this model works adequately will depend on its total capacity compared to total demand, and in turn this will depend on available funds compared to data set size.

The fourth problem is probably best solved by standardizing on one or, at most, two commercial database products. The cost and licensing structure for such products means that a common choice should be made across DØ , CDF and the Computer Division.

In devising a DØ plan specifically intended to avoid the above problems a number of strategic guidelines, listed below, were generally applied.

1. Use modern mainstream technology for both hardware and software whenever possible.

2. Internally develop components of the computing system only when no acceptable freeware or commercial options exist.

3. When development projects are required, attempt to share the technical effort – with CDF and the Computing Division; with other HEP and NP projects; with the computing community in general.

4. Design systems which are highly modular and layered to ease testing and debugging and to facilitate the introduction of improved components and new technology.
5. Design system components to established standards – public standards when possible, project standards otherwise.
6. In addition to price/performance, the reliability and maintainability of a component should be seriously considered in evaluating it for inclusion in the computing system.
7. Design systems which minimize the movement of data.

As already discussed briefly, the scale of the software development effort for Run II is quite comparable to that of Run I. In Run II, the system will again include multiple platforms (at least three currently supported flavors of UNIX, and very likely some version of the NT operating system as well, by the end of Run II). It will once again be necessary to deal with new detector subsystems, and therefore with rapidly changing software as new algorithms are implemented. An important priority in the Run II design must be to obtain better performance in the areas of maintainability, portability, and verification. While the number of physicists in the DØ collaboration is a factor of 2-3 larger today than it was when the software system for Run I was being developed, there are two reasons why this does not reflect itself in an ability to apply significantly more technical effort to the problem. First, interesting data sets from Run I are absorbing a significant fraction of the attention of many of the more software oriented collaborators. Second, the time available for the development of the Run II software, three years, is approximately one half the time taken in developing the software for Run I. Since the amount of available technical effort is a limiting resource, perhaps the most limiting resource, it is critically important that it be used most effectively. Recent developments in software technology are explicitly directed toward increasing the efficiency of technical effort in producing high quality software, so DØ has undertaken an investigation of such new technologies.

In particular, since the issuance of DØ Note 2537 in April 1995 (*DØ Analysis Computing Recommendations for Run II*), DØ has been actively pursuing the use of object-oriented design and languages for its Run II reconstruction system. A great deal has been learned from these investigations and from various prototyping activities. The conclusions to date are the following:

1. Object-oriented design technology is clearly directed at solving exactly the problem faced by DØ of putting together a large system from pieces which are separately testable and easily maintainable by a large team of programmers. OOAD is not, however, a panacea. Its use certainly does not *guarantee* these features; it does, however, supply the tools to achieve them. In particular, the OOAD design tools now available, such as RationalRose, GraphicalDesigner, and PC development environments, offer far better ability to keep a program model up-to-date and to make changes to either code or model that propagate from one to the other. If nothing else, this provides an automated documentation tool far superior to what DØ has had available in the past. It should be possible to get far more out of the design process than just good documentation, however. With training and experience, it should be possible to critique these designs to see whether they meet the requirements of modular testability, before writing a line of code.
2. The training referred to in the previous paragraph is an essential component of moving to this new technology. There is little time and a great deal to learn. Formal training in OOAD and C++ provides a much-needed jump start to those seriously involved in the reconstruction effort right now. DØ has sent almost 40 people who will definitely be involved in our Run II software through one or more of the Fermilab-sponsored courses in

OOAD and C++. (It has also trained another 25 or so students and postdocs who may or may not remain on DØ.)

3. The major pain in Run I computing and software came when being forced to change major system components, either hardware or software. Some components were simply too interwoven to come out, as for example the ZEBRA system. The lessons learned there have suggested that the basic philosophy for Run II should be to try very hard to put together *all* components of the system in such a way that replacement is never very hard. This suggests that a mixture of functions such as the ZEBRA system contains, for example, should not be allowed. The memory management function should be handled by the language itself (implying a modern language like C++ or Fortran 90), while the mechanism for persistent storage across platforms should be something less integral to the program, which can be replaced. The current plan is to use the DSPACK system for persistent storage, but NOT for memory management. The interface to DSPACK is being explicitly written to facilitate changing to another persistent storage mechanism (e.g. an object-oriented database) later if so desired.

4. The decision to utilize C++ places DØ on the modern side of the dividing line between the experiments for whom the end is in sight (the fixed target and LEP II experiments) and those which expect to begin in or continue into the 21st century (LHC experiments, BABAR). While this decision clearly involves more up front work, and more risk, it also gives far more options in the medium term future to utilize both commercial software and the LHC software as it comes online. It is far more consistent with the basic Run II philosophy enunciated in the previous paragraph.

1.4 The Run II Computing and Software Organization

The formal organization to manage Run II computing and software was put in place in March 1996. The co-leaders of the organization are Bruce Gibbard and Wyatt Merritt. The organization is divided into four groups, each with two co-leaders. The co-leaders of those groups, along with Gibbard and Merritt, form the Computing Planning Board which meets biweekly and is expected to oversee progress in general, to insure communication between the four groups, and to resolve issues that cross group lines of responsibility. The four groups and their charges are:

- Software Tools Group (Herb Greenlee, Harrison Prosper). This group is responsible for the software development environment (e.g. compiler and CASE tool choices), for software tools required in the system (e.g., databases and GUIs), for graphics displays for the experiment, for the generic framework program(s), and for the object persistency mechanism.

- Online Computing and Software Group (Stu Fuess, Bruce Gibbard). This group is responsible for the online computing system, which will handle the data acquisition, control and monitoring of the experiment, and the Level 3 trigger framework software.

- Algorithms Group (Serban Protopopescu, John Womersley). This group is responsible for designing and coding the algorithms for the reconstruction program and the Level 3 trigger, and for the simulation programs that model the detector.

- Production and Data Access Group (Drew Baden, Mike Diesburg). This group is responsible for planning and operation of the reconstruction farm, and for the design, procurement, and maintenance of the systems with which we access and analyze the reconstructed data.

2 The Software Development Environment

The Run II organization has one group, the Software Tools Group led by Herb Greenlee and Harrison Prosper, devoted to providing and maintaining the software infrastructure needed for the development and run-time environments for our online and offline code. This group also provides those applications which are specific to our domain, but require mainly computer science rather than physics knowledge to implement — graphics displays, databases, and configuration management procedures are the primary examples of such applications. The system used for object persistency and cross-platform data transport is also the responsibility of this group.

Following the philosophy enunciated in Sec. 1.3 above, we have tried to use free or commercial products for these tasks whenever possible, and to use general standards and to design in the possibility of replacement for any component without excessive pain. The Software Tools Working Group (led by Qizhong Li), its several subgroups, and in some cases joint working groups with Computing Division and CDF participation have been the agents for making recommendations about the products we should use, maintain, or develop. This recommendations phase is nearly ended, and we are moving into the phase of implementing support and development for the chosen products.

2.1 Languages, Libraries and Standards

During the latter half of 1995, the DØ Languages and Data Structures committee met to study issues related to choice of programming languages, design methodology, and data persistency mechanism. The recommendations of this committee [1] are still guiding current DØ run 2 software efforts.

One basic recommendation of the Language and Data Structure committee was a qualified endorsement of object oriented analysis and design (OOAD) techniques in general and the C++ programming language in particular for most run 2 software development efforts. The committee also recommended that DØ support bilanguage software development using C++ and Fortran-90, with Fortran being limited to major numerical and algorithmic software components (if justified) and legacy code. In the time since the committee made its recommendation, the transition from Fortran to C++ and OOAD has acquired greater impetus, because of the focus on design issues for the new system and the better design tools available in the C++/OOAD world. Areas of active software development are now mostly focusing on OOAD and C++. As yet, there has been no bilanguage software development in DØ, but preliminary investigation of the technical issues and design compromises that would be involved indicates that a mixed reconstruction system is definitely not a desirable option.

The transition from Fortran to C++ is also being supported by the laboratory through the joint C++ working group, with membership drawn from CDF, DØ, and the Fermilab Computing Division [2]. The activities of this group include the following:

- Development of a set of requirements and standards for the evaluation of C++ compilers and debuggers.
- Development of a suite of test programs for evaluating compliance of compilers with newer features of the draft ANSI C++ standard.
- Evaluation of existing C++ compilers.
- Recommended language feature usage and coding style guidlines.

The C++ working group has published a set of interim C++ compiler and language feature usage recommendations [3]. The recommendation with the most impact on DØ is the choice of compiler, which is gnu C++ (g++) for the platforms common in DØ (sgi and ibm), rather than the vendor supplied compilers. As yet, none of the currently available compilers for the platforms in wide use at DØ, including g++, support the full draft ANSI standard, and this decision needs to be subjected to periodic review as the various compilers mature.

C++ software development will require a set of class libraries to replace the functionality that had formerly been supplied by the CERNLIB subroutine libraries. At present there is no agreed on set of class libraries that is suitable for multiplatform software development beyond the STL class library, which is included in the draft ANSI standard. To address this need, the Fermilab Computing Division has agreed to sponsor another joint working group, the "Fermi Physics Class Library Task Force," [4] with members from CDF, DØ, the Fermilab Computing Division and the Fermilab Accelerator Division. This group has been charged with the actual development of a physics class library for laboratory-wide use. The group held a workshop on Jan. 13-14, 1997, to define initial specifications and priorities for the FPCL, and working assignments in accordance with those priorities have been handed out. This group is expected to keep in close touch, and share effort where possible, with related HEP enterprises such as CLHEP and LHC++.

2.2 Object Persistency Mechanism

It has been recognized for some time that DØ will have to replace the Fortran based, CERN supplied ZEBRA data management system. The greater part of the efforts of the DØ Language and Data Structure committee were spent studying this question. The committee produced a recommendation [1] that is still being followed in current development efforts. The basic recommendation of the committee was that DØ adopt DSPACK, a data handling package developed by the NA49 experiment at CERN, as the ZEBRA replacement.

The committee pointed out that DSPACK should be used in a different way than ZEBRA has been used by DØ in Run I. ZEBRA actually provides two functionalities: Memory management and platform independent data I/O. The committee recommendation, and the current plan, is to separate these two functionalities, with the former being supplied primarily by the language (although DSPACK does have its own built in memory management capability). Because of its use for memory management, ZEBRA became so thoroughly embedded in DØ software in Run I that changing later to a different system was out of the question. The current plan calls for having no naked DSPACK calls in user written software.

After the Language and Data Structure report was issued, a working group was formed to continue to develop a data persistency mechanism based on DSPACK. The DSPACK group is currently actively engaged in the design and implementation of a C++ persistent object interface. The design of the C++ interface for persistent classes is at an advanced stage [6]. The C++ interface is loosely based on the ODMG standard C++ programming interface for object oriented databases [5]. The ODMG standard defines a standard object model for all persistent objects, which specifies which data types and classes can be composed into persistent classes, and how they can be composed. The object model proposed by the DSPACK group [6] follows the ODMG object model closely, but with different names for the ODMG specified classes, to avoid possible conflicts with other ODMG compliant object databases. The ODMG standard also specifies a standard I/O interface. The current design does not follow the ODMG I/O interface, but has its own interface based on a more event-oriented I/O model.

The features of the persistent class design are as follows:

- Object definition language for specifying persistent classes that can be preprocessed into source and header files.
- Base class for all persistent classes.
- Smart pointer and relationship classes to allow transparent physical partitioning of events.
- Template collection classes.
- Event oriented I/O interface for sequential and random access I/O.
- Support for the possibility of multiple I/O mechanisms (besides just DSPACK).

In this model, DSPACK is used only for its ability to "flatten" complex data structures into something that can be written and read. It is forseen that we may want to use some other I/O mechanism besides DSPACK, if not for event data, then certainly for calibration data. One likely possibility is the use of a true object-oriented database for calibration data and other typical database usages.

2.3 Framework Builder

The software framework provides a common environment for physicists writing data analysis software modules. The common environment makes it easier for physicists to both write and reuse code. DØ used a framework in Run I to great effect; the physicist was given a standard set of software packages that performed jet corrections or electron finding that could be plugged into the framework along with the physicist's private analysis module.

The Run II framework will perform the same function. The same framework, with modifications, will be used in the following environments:

- *Physics Analysis:* Good debugging tools and a user interface are required to help the physicist here, as well as being able to run in a batch interface.
- *Calibration:* Special types of event records contain calibration data and must often be handled in a special way.
- *Online Monitoring:* Keep track in real time of how the detector is performing by, for example, looking for dead or hot channels or by attempting to reconstruct W bosons.
- *Reconstruction:* Full event reconstruction often takes place in a farm environment, fully controlled by script systems.
- *Level 3 Trigger:* The online software trigger must be fast and efficient.

If possible, we would like to meet these requirements with a common framework, at least to the extent that code implementing physics algorithms can be moved from one environment to another. It should be possible, for example, to develop a new software trigger tool offline, and move it into the online environment rather having to rewrite it. However, the requirements for the various environments are sufficiently different that it may not be possible to completely unify them all (the Level 3 trigger, for example).

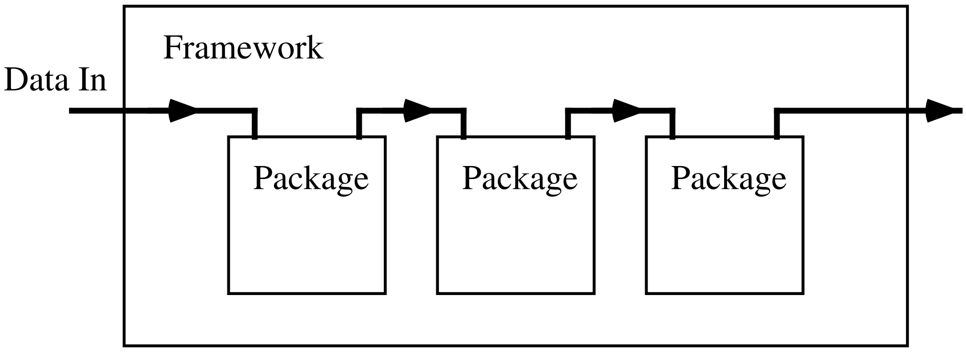


Figure 1: The script system will be the glue that holds the framework together. The script system will be responsible for running individual modules in order and passing event data between them.

2.3.1 The Framework Design

DØ has made the decision to move all large software projects to C++, including the framework. There are two approaches to a project in C++. First, the *as needed* approach in which you generate the data and objects as they are required by the program. For example, a list of jets is built from calorimeter towers only if the list is required by the analysis code. The second approach, the *framework* approach, has a set of modules that execute sequentially, each with a specified task. Data moves between the modules in a common format. In this approach, the list of jets is generated for each event whether or not is is needed. Each approach has its advantages.

One of the key requirements for a software effort this size is that physicists be able to trade code. In this respect the framework has an advantage as it imposes a common interface on everyone's code. In addition, it becomes simple to compose modules from separate sources without having to modify any of them. It may mean more work in the short run to fit the code into the common interface; in the long run, however, we believe the reusability will more than pay for the effort.

There are some aspects of the *as needed* approach which are attractive, however. Code often requires certain types of data as input. In the present Run I software, this is dealt with in a variety of *ad hoc* manners, sometimes reporting an error, sometimes calling a detector-specific reconstruction routine to generate the data. This had the effect of generating a profusion of poorly controlled interdependencies between modules, and often resulted in users having to link large amounts of code into analysis programs which was never used. In Run II we plan to allow one software package to invoke another, but in a controlled manner. Furthermore, resolution of the calls will take place at run time rather than at link time, giving the end user better control over what code is actually linked into analysis programs.

The framework approach is not as flexible as the *as needed* approach. To increase its flexibility we plan to make the glue that holds the individual software packages together an interpreted script system. The job of the framework glue is to guide the data between the various framework modules; not, normally, to do any processing. For this reason we have chosen to use a freely available script system to serve as this glue. Figure 1 shows how we envision the script system and the software analysis packages working together. Data can move through the packages differently depending upon the environment. The script system will provide a quick way to make these changes without having to alter the individual software package interface.

Each analysis package is potentially configurable by a number of parameters used to adjust thresholds like jet cuts. In Run I we used files that contained lists of parameters and their value.

In Run II we plan to use a similar mechanism. As in Run I, we will use a file that contains list of parameters, but, they will be in the script system's format. This will allow for intelligent setting of parameters. Further, the user interface will allow for the setting of parameters at run time, overriding the external parameter lists. This interface will also enable one to invoke special code in a module (a dump routine,for example).

The different environments that the framework will operate in demand different levels of user interaction. No run-time user interaction is required by the Level 3 trigger framework or the production reconstruction jobs, but a very interactive interface is desirable in debugging a physicist's analysis program. The user interface will be flexible enough to adapt to the different possible running conditions.

2.3.2 Software Required

A number of software components are required to make this system work. In part, they are required because the average physicist should not, and has no wish to, have to understand the details of the framework's inner workings. To this end a *Program Builder* will be written. The program builder takes a set of known framework packages and produces object files and parameter lists and, possibly, an executable to run data through the list of specified modules.

The *user interface* for the various frameworks must be written. This may range from a fairly complex GUI interface for the user analysis framework to something much simpler that depends on the parameter passing mechanism for reconstruction, where no human interaction is possible.

The *framework* must also be written, of course. The code here is responsible for interfacing the various frameworks together, generic framework code, interfacing to the script system, etc.

2.3.3 Current Status

A prototype of the system is now available:

- Prototype framework code is ready. This includes the glue and a parameter style user interface using the Python scripting system.
- A very crude version of the program builder that will stitch a list of modules into an executable, produce link scripts, and generate a top level python system to run the whole thing.
- A fairly sophisticated parameter list system based on the python script system. It is capable of reading the old and a new style of parameter lists.
- I/O modules to read the Zebra based data files used in Run I.

The user interface for the December 15 version of the framework is very basic. The final version of the framework will contain a user interface based on the Tk toolkit, a public domain X-windows toolkit, that has been interfaced to the script system. There will also be a command line and batch version of the interface.

The second version of the framework is due ninety days from Dec 15, in March. It is expected that the framework will evolve more slowly after that.

2.4 Databases

An initial evaluation is in progress to determine the Database Management System (DBMS) needs and solutions for Run II. In order to understand these needs, the uses of DBMSs in

Run I have been re-examined and the potential needs of DBMSs in Run II considered. Several requirements have been established which should be met by any DBMS which we might use. Preliminary research into potential commercial and public domain products which might be suitable to our needs has been done [7]. Evaluation and testing is proceeding for the most interesting products reviewed.

During DØ Run I, several DBMSs were used to manage a wide variety of information. They include DBL3, RZ, DEC Rdb and Oracle. DBL3 is a CERN database product which provides keyed access to information using the ZEBRA RZ file structure. It was used for managing detector calibration and monitoring information which was stored online, compressed, and distributed to the offline processing cluster. This product was chosen in part because of its platform independence. FATMEN is a hierarchical data management system used to catalog the raw event data and track the offline data processing. It employs the RZ file structure to provide keyed access to stored information. DEC Rdb, a commercial relational DB product from Digital Equipment Corporation, was used for the Tape Log DB, Hardware DB and Production DB. The Tape Log DB kept track of all run, trigger, filter, raw file and tape information taken online. The Hardware DB was used in the online system to provide device access information to the front end system for control and monitoring purpose. The Production DB managed all the run, trigger, filter, file and tape information for all runs and processed data for Run I. It also managed all the luminosity information for the runs. Oracle, a relational database designed by Oracle, Inc., has similar features to DEC Rdb and was used to track the DØ personnel training information during the run.

There are many needs for information management during the next running period as a wide variety of information will be maintained. Calibration, monitoring, luminosity, run summary, trigger, filter and event information will need to be stored when data taking resumes. Access information for hardware devices used in online control and monitoring will also be needed. Offline, many of the databases written during running will need to be accessed by the reconstruction farm processes. Data processing (Raw, STA, DST etc. or logical equivalents) and physical location (file) information will be maintained for all event data. Not all of the databases that we used in Run I should be used in Run II. The FATMEN Catalog and the Production DataBase should be merged into one database. Some of the databases which were not utilized in Run I (i.e. the luminosity DBL3 database) will be eliminated. Also DØ has chosen to move away from ZEBRA so all the DBL3 databases will need to be replaced. Hopefully, this will lead to a more reliable and easily maintained solution. In addition all of the Rdb databases, which are VMS specific, will also be replaced with platform independent databases.

Requirements for a Run II DBMS can, at this point, only be scaled from our Run I experience to the more complex and higher event data rate anticipated for Run II. Having a single DB product to serve all the Run II needs would simplify the effort required to train personnel and manage the information. Due to the complex nature of the data to be stored, it appears that an Object-Oriented DBMS is best suited to our needs. A commercial product would provide a more reliable service and higher quality tools for access and performance monitoring than public domain software. The requirements and preferences for such a product include: 1. Allow automatic replication(copying) of a database across DØ supported platforms, 2. Support Object and Schema versioning, 3. Meet performance requirement needs for Run II, e.g. Online calibration data must be accessible no later than 5 minutes after a calibration run has finished, 4. Robust and corruption free performance, 5. Performance must scale with the expected number of users and size of the database (the database may be replicated to help improve performance), 6. Work in a fully distributed, multi-platform environment 7. Be standards compliant (such as ODMG, SQL, C++ for Object Oriented databases), 8. Must have sufficient database administration tools for easy setup and optimization and, 9. Provide monitoring capabilities. In addition, the licensing must be affordable, sitewide and there must be ways to include remote institutions. We are also likely to need query access via the Web.

An additional minimal database solution may be required if the database chosen cannot meet the performance requirements necessary for the online calibration and offline reconstruction. For example, in Run I Rdb was not used for the calibration database because it could not be accessed from a UNIX platform and DBL3 was used instead. However, the reliability and tool support for DBL3 was inferior to most commercial products and considerable effort was required to overcome this. If a minimal database solution becomes necessary, the product should also meet strict requirements to avoid these problems.

We have explored several commercial and public domain database products in the context of Run II usage. The databases reviewed can be categorized into three different types: relational (RDBMS), object-relational (ORDBMS), or object-oriented (ODBMS). A relation database is tabular in format and uses index tables to allow for one-to-many or many-to-many relationships. An object-relational database can be a true object-oriented database that can mimic a relational database (has SQL access) or it can be a relational database with C++ wrappers around it. An object-oriented database stores its information in object format. This means that one-to-many and many-to-many relationships are part of the object (via a pointer) instead of separated from it. The following table summarizes how each product reviewed is categorized and by whom it is supported.

Product	Support	Pub.Dom./Comm.	Type
DEC Rdb	Digital Equipment Corporation	Commercial	RDBMS
Illustra	Illustra, Inc.	Commercial	ORDBMS
MiniSQL		Public Domain	RDBMS
O2	O2 Technology, Inc.	Commercial	ODBMS
Object Store	Object Design, Inc.	Commercial	ODBMS
Objectivity	Objectivity, Inc.	Commercial	ODBMS
Oracle	Oracle, Inc.	Commercial	ODBMS
POSTGRES		Public Domain	ODBMS
SHORE	University of Wisconsin	Public Domain	ODBMS
UniSQL	UniSQL, Inc.	Commercial	ORDBMS
Versant	Versant Object Technology	Commercial	ODBMS

After careful consideration four DBMS candidates were found to meet our needs and for various reasons the others are removed from the list. The products which meet all the criteria are UniSQL, Objectivity, Object Store and O2; these systems will be subjected to further testing. The other products were removed from the primary list for various reasons and include MSQL, SHORE, POSTGRES, DEC Rdb, Oracle, Illustra and Versant. MSQL only allowed single key access (can only access by run or time but not both) and does not commercially supported. SHORE and POSTGRES are not commercially supported products. DEC (Oracle) Rdb and Oracle are relational in nature and, while they served us well in Run I, an object-oriented solution is better (pending testing of the ODBMS solution). Illustra was removed because it is not a true Object Database or even a hybrid but rather a relational database with C++ wrapped around it. It also has just recently been purchased by Informix and will be undergoing major modifications over the next several years. Versant was removed because it did not provide enough basic database tool support. It should be noted that any of the databases removed could be considered for a minimal solution if the testing of the ODBMS shows that a minimal solution is necessary.

The commercial database solutions which have been chosen as candidates are, of course, not free. Most of the companies included in the final list have tiers of pricing based on developer, server, and runtime categories. The cost of licensing these products depends on our needs but we can make some rough estimates that the costs for DØ may range around $ 200K. Most of these companies are willing to make special concessions to the lab and provide special rates for experiment or site licensing agreements. Some of the companies understand the situation with University collaborators as well and are willing to include them as long as their use of the products is DØ related. Cost and licensing are certainly important issues and will continue to be addressed along with the technical issues involved with the products.

Evaluation and testing has begun to further understand how particular ODBMS products might meet our needs; this work is proceeding jointly with CDF under the sanction of the Computing Division. As a first step, transferring part of the present Production DataBase information to an ODBMS will allow performance testing and comparison with existing tools used in run I. As example applications are put together designs for various db need areas will begin to develop and these can also be used for testing. We will likely run benchmarks on various evaluation products, perhaps using a suite used by the OO7 project at the University of Wisconsin. The prospect of using an ODBMS for data persistency is also being explored. To date, an evaluation copy of Objectivity/db is in hand and in the future there are plans to also evaluate the other primary candidates which emerged from the database review.

2.5 Script Languages

DØ will need to use one or more scripting languages in Run II to provide the functionality of run-time input to programs. In Run I, there were several mechanisms for such input: RCP files for parameter input, Program Builder files for program link control, DCL files for library release and program execution control. Ideally, one could perform all these functions using a single cross-platform scripting language, choosing from several freeware products available. A subgroup of the Software Tools Working Group was formed to consider this issue, and has now recommended that the language Python be used for all scripts with the exception of the following cases:

1. New scripts for applications in which all of the existing scripts are written in some other language.
2. Applications for which some other language has a significant advantage. For example, perl when extensive use of regular expressions is required.
3. Very short shell scripts.

In this recommendation **??**, the committee explains its exhaustive inquiry into the capabilities of Python relative to the other choices (Tcl/TK, Java, perl, etc.) and also into the issue of language support. They concluded that Python was the best choice on purely technical grounds; they also concluded that the strong user community for this language was sufficiently committed to its support to insure its availability for the duration of the Run II needs. It is currently supported on Unix and Windows NT (and other operating systems not of interest to us).

2.6 Graphical User Interfaces

Many Run II graphical user interface needs will exist in DØ and several options exist for dealing with them. These options have widely varied requirements for programming skill, product performance and functionality. An array of tools exists for creating menu-type GUIs using a number of simple languages generically referred to as "scripting" languages; these are attractive and may fulfill the needs of many of our applications. It is clear that there will be Web-based GUIs which will provide information about the operation of various systems, data processing and data access. These needs might best be served by recently developed technologies which provide a range of tools specifically targeted at such problems. Some of our applications may prove too complex for simple solutions and will require detailed knowledge of more intricate lower level tools. This may be the case for some on-line applications and possibly specific off-line areas, such as event display.

The criteria for choosing the tools to employ are dependent on the application, but in general we will need:

1. ease of use
2. ease of programming
3. platform independence (at least, portability to Unix and NT)
4. longevity

For most of the applications, a scripting language connected to a graphics toolkit such as Tk seems to provide the most flexible solution. A separate subgroup evaluated scripting languages and recommended the use of Python, as detailed in the previous section of this document.

It is planned that prototypes be created for some applications. While pursuing this, a couple of different scripts written in perl and Python might elucidate the differences between these solutions and provide additional information on the types of difficulties encountered with each. Included in the prototypes might be the following examples:

1. A set of tools to assist in guiding users through the steps of code release.
2. A script using named pipes or socket communication with a user application; this is one arrangement which will be used frequently.

There may be instances which the scripting language solution does not fit, and a full blown X/Motif or other effort is required. Hopefully, these cases are few, but we will learn with more experience. We will also keep our "ear to the rail" for new attractive solutions.

2.7 Graphics

Beginning in the Spring of 1995, the DØ Run II Future Graphics Working Group started to gather information on how to develop and improve the use of 2D and 3D event graphics in DØ, in order to capitalize on the modern graphics hardware which would be available by the beginning of Run II. Its initial work involved talking to and polling DØ graphics users; studying the graphics in use by other collider experiments–here and abroad; holding periodic working meetings of the group, and attending graphics conferences. HEPVIS95, in particular, turned out to be a meeting where many of the current approaches were solidified. In spring 1996, the working group became the Run II Graphics group, led by Mike Shupe and Saul Youssef.

It became apparent that improvements are needed in two general areas:

- Two dimensional displays must be more carefully planned with regard to visibility and information content, and must be more interactive.

- Three dimensional displays must rise to the standards of modern high-end workstation graphics, with improved modeling and the use of sophisticated viewers.

There are existence proofs for both of these areas. The ALEPH displays written by Hans Drevermann are perfect examples of 2D visibility and interactivity. The SCIAN based DØ displays created by Saul Youssef show the power available in modern 3D imagery. Given these examples, the working group focussed on tactics for having full capability in both areas available to the widest possible DØ audience.

There are a large number of factors which we are trying to take into account when working out implementations of the above goals:

- Graphics languages and protocols are rapidly changing. Among the public domain standards, and those which tend to come as bundled software on workstations, we have PHIGS, OpenGL, VRML, JAVA, straight Xlib graphics (2D), and many others. Just recently, SGI announced Cosmo-3D, which appears to be an improved superset of OpenGL. For direct writing of graphics from drawing modules, the working group prefers OpenGL because of its power and platform independence. But the real lesson we took from all this is that we must use a thin layer of graphics entry points because the software must persist for many years even as new graphics standards appear at the rate of one-to-several per year. To first order, new standards may then be incorporated quickly by simply writing a small "driver" module.

- Commercial graphics software packages, such as Open Inventor, have the advantage of providing many powerful 3D graphics operations. At the same time, they may be too costly for some university groups. After the Run 1 experience with DI3000, the working group was cautious about committing completely to the use of a commercial package. One tradeoff against these costs, for example, might be to rely more on central graphics servers with remote graphics viewers. Such a model then depends on having good network bandwidth to get reasonable response when viewing in 3D.

- Graphics hardware is a related issue. DØ at Fermilab and at the universities will not be replacing its hardware overnight. For this reason, development must be done not only on high-end workstations, but also on devices understanding only basic protocols such as X-Windows. The "driver" model mentioned above accommodates this well.

- There is a particular catch with graphics: to be useful, the graphics should come into being early, so that it is available for algorithm debugging. But many pieces of the environment in which the graphics lives are not yet developed: the geometry database, the run and event data structures, and the frame. For this reason, graphics modules should employ isolating entry points, analogous to the graphics thin layer, to allow "scheme independent" access to event and geometry data. As the environment develops, the code behind the entries can change.

These considerations, and more, have led to the current evolution of Run II graphics work in DØ. Below we discuss progress and plans on a project by project basis.

2.7.1 STARGAZER

Inspired by the recent renaming of the Run II Monte Carlo package UPG_GEANT to "DØGSTAR," the graphics program being developed as a replacement to PIXIE is now called "STARGAZER." STARGAZER is written in Fortran, and is designed to be machine independent. It currently runs on DEC VAX and Alpha machines under VMS, and on SGI workstations. Soon it will be tried on PC's running Windows NT, with the Microsoft Fortran development system. The current configuration, capabilities, and status are as follows:

```
1.  Screen graphics:  X-Windows, GL, OpenGL       (working)
                      Windows NT (QuickWin)       (to be debugged)

2.  Output formats:   Postscript (vector, color)  (working)
                      HPGL2, VRML, NFF            (to be provided)

3.  Displays:         Calorimeter side, end, 3D and lego displays.
                      SciFi hits, clusters, spacepoints, and tracks.
                      Silicon spacepoints.
                      Muon A-Phi, pixel, and MDT hits.
   (Being developed:  Central and forward preshowers, WAMUS, more
                      displays for all subsystems.)

4.  Current GUI:      Standalone viewer written in same language as
                      the displays.  Includes:  spin, zoom, move, grab,
                      fisheye transforms, etc.  Multizone windows.
                      User customized windows.  Hardcopy control.
   (Being debugged:   Object visibility flags.  Zone jumping.
                      Rubberbanding.)
```

STARGAZER is beginning to be used by people doing algorithm development. The near future plans are:

1. To roughly double the number of basic displays. At that point there will be an initial usable set for each subsystem. These displays will then constitute working "examples," so that specialists in each subsystem can jump in and do display development. The learning curve should be short, because the graphics modules are all purely "drawing" routines, creating shapes in 2D and 3D. Viewing transformations and the formation of complex displays are all handled by the STARGAZER framework.

2. To complete the screen drivers, output drivers, and GUI features listed in the section above, and prepare the code for long-term library maintenance.

3. To subject the STARGAZER package to several development projects: (1) a study of the SciFi tracker options XUV-X vs. XU-XV (with Norman Graf and John Womersley), (2) to optimize the visibility of the graphics for the new muon system detector and trigger elements (with Ken Johns and Ajay Narayanan), and (3) to allow input from the McFIO package, also in use by the McFIV project described below (with Norman Graf).

The longer term plans for STARGAZER are:

1. To connect to the DØgeometry database and event data structures as they are finalized and filled with information.

2. To seek comments and suggestions from DØ collaborators using STARGAZER and to undergo cycles of improvement. To involve people, as they become interested, in creating new DØ Run II displays.

3. To add new drivers as new graphics standards appear (if they make STARGAZER more universal or improve performance).

2.7.2 McFIV: Open-Inventor with McFast

At the time of HEPVIS95, the working group was considering the use of scientific visualization frames such as Explorer, SCIAN, and AVS. But deeper study of these frames showed that making a tight connection to the detector geometry description and data structures is very difficult by this route. And without such a connection, interactive graphics in the "ALEPH style" is not possible. Nevertheless, there had been very successful 3D DØ displays done in the SCIAN visualizer (by Saul Youssef), and in the Explorer visualizer, using Open-Inventor (by Jeff Kallenbach, Paul Lebrun, Greg Griffiths, and Amber Boehnlein), showing that the use of these tools could provide powerful 3D manipulation without a lot of programming.

In the past year, a breakthrough occurred when Amber Boehnlein began using Open-Inventor directly, employing its own windowing system and graphics manipulators in the McFast frame, and avoiding the Explorer framework. This tactic has restored the possibility of making a very direct connection to the geometry and data.

The Open-Inventor graphics modules are written in C++, and machine independent I/O is handled via the McFIO package. The McFast and McFIO package are supported by the Fermilab Computing Department. The advantage of this approach is again, that much of the complexity is taken care of by the Open-Inventor system. The disadvantages are the cost of Open-Inventor (although the cost of the visualization frame is being avoided) and, possibly, less machine independence. The project is available on non-SGI platforms, but has not yet been evaluated. Nevertheless, the consensus is that the advantages are strong enough that this path should be pursued along with STARGAZER.

The current status is:

1. Displays exist for hit calorimeter cells in 3D, η projection, and ϕ projection. Tracks can be displayed in the same views. The DØ silicon barrels and disks can also be represented.
2. The data are created by running McFast and writing out an McFio file.
3. The Open-Inventor manipulators in use so far are the Examiner Viewer, an Open-Inventor class which allows the manipulation of a 3D scene. The user can set and return to a home view, turn off perspective, and bring the full scene into view. Picking may also be enabled. Rotations and translations are performed with the mouse or with dials. The viewer also allows the user to change from solid to wireframe representation and to change the transparency model. No code was written by Amber to obtain this functionality.
4. The GUI is Motif based.

Near future plans are to recruit another person at Fermilab to work with Amber. Fred Borcherding and Bob Angstat have expressed some interest, but they are quite pressed for time and are interested primarily in the fibers. An additional person (.25 FTE) is necessary.

Longer term plans are:

1. To connect to the DØ geometry database and event data structures as they are finalized and filled with information.
2. To seek comments and suggestions from DØ collaborators using McFIV and to undergo cycles of improvement. To involve people, as they become interested, in creating new DØ Run II displays.

2.7.3 CTool

In addition to McFiv and STARGAZER, upgrade event displays have been available within the CTool package for some time. You can see examples of these displays in the www "Event Display Staging Area" under the "Physics Analysis" page and a description of how it works in the HepVis proceedings (http://fnpspa.fnal.gov/workshop/talks/index.html). Because of its great flexibility in what can be displayed, CTool is particularly useful for debugging simulation software and for tracking down unusual detector problems. Although one currently needs SciAn (GL based) and view the CTool displays, it would be an easy operation to absorb CTool displays within, e.g. STARGAZER or McFiv.

To sum up, we have three active event display projects which have already achieved a substantial level of operation. We plan to continue in this way for approximately one year while the detailed framework for Run II software is worked out.

2.8 Configuration Management

The Configuration Management System that is now taking shape for Run II DØ owes its genesis to several preceding groups. In March of 1995 the Data Structures and Languages Committee laid out, in broad outline, the environment in which we would be working for Run II [9]. In May the DØ Software Tools Group was formed to recommend solutions to the many problems arising from the changes in that environment. Among these problems was the entire question of Configuration or Code Management. By March of 1996 the Tools Group had agreed upon a minimal list of requirements that any Configuration Management system to be used by DØ must meet [10]. The salient points in Ref. [10] are:

- Must support multiple hardware platforms, Unix of various flavors in particular.
- Must support multiple languages, C, C++, Fortran and Fortran 90 as a minimum, as decided by the DØ Languages Committee.
- Must provide several "tagged" versions of each library (Packages) to be available at all times.
- Product versions must be available that tie together compatible package versions. But it must be possible to override particular package versions within a product version although recompilation may be necessary.
- Official releases and their verification must be done centrally to a small number of platforms having well defined environments. These releases would then be available, with their "automatic" build files for export to remote sites.
- Distribution should be done by "pulling" from the remote site in several modes depending on the level of operating system compliance at the remote site.
- Test releases of the individual packages must be able to be done by the library "czar."
- Code verification should be done in several steps, from ascertaining that the code compiles on all target platforms to full verification of results using simulated and real data. The envisioned steps were outlined.
- It must be easy/possible to retrofit all versions of interest with bug fixes.

In March of 1996, a Joint CD/DØ/CDF Code Management Group was formed to recommend common solutions to the Configuration Management problems that would be applicable to both collider experiments and, hopefully, to the laboratory at large. By May 1996, that group had issued a recommendation for a set of tools that, it was felt, met the requirements listed above, among others [11].

Shortly thereafter the Joint Group was reconstituted with additions in membership into a working group to implement the earlier group's recommendations. As often happens, once implementation began, new problems and solutions appeared. So the current (partial) implementation is not precisely that outlined in the Joint Group's recommendations. A good, fairly detailed description can be found in "A UNIX Based Software Management System" edited by Robert Harris of CDF [12]. In addition, a (draft) "how-to" note [13] gives directions on how to get started using the system at DØ, with many references to the primary online documentation.

As currently being implemented, the Configuration Management System to be used consists of: Concurrent Versioning System (CVS) with remote extensions including some developed by the Sloan Digital Sky Survey at Fermilab; the release directory structure and modified versions of the release procedures developed by the BaBar experiment at SLAC; and UPS/UPD from FNAL to do the code distribution.

CVS is similar to DEC's CMS (which we have been using for Run I software) with two major differences:

- CVS is directory based. Entire directory trees are stored and retrieved, rather than being flat with groups like CMS.
- CVS promotes "concurrent" development of code. Its control mechanism relies on merging concurrent changes to files rather than the CMS tactic of linear development by reserving files. Each system has advantages and disadvantages and both require care and communication on the part of and between developers. Neither guarantees that code changes won't be overwritten.

The remote extensions to CVS developed in part by the Sloan Digital Sky Survey group at Fermilab solve most of the file ownership and access and security problems of native CVS and allows development at many remote sites with all code centrally stored in a uniform an transparent manner.

The BaBar release structure solves several problems in a rather elegant way. Most notably it helps to solve the "coupling" problem between packages caused by header include files and shared code. For details see Ref. [14]. This structure provides two main branches in the distribution tree:

- Packages: contain the source code from CVS corresponding to particular versions of the code as determined by the individual library's czar and released, at will, by the czar. There can be many versions of each package, all independent.
- Releases: contain consistent combinations of all packages needed by a product. The package versions are included by symbolic links to a particular version of a package in the package area. This area contains all executables and object libraries used by the release. Only the release area contains compiled binaries, thus solving the tight coupling problem aggravated by C and C++ header files and other include files.

The BaBar release procedures work with the release directory structure to allow a developer, or any user to link with and develope code for any package(s). In essence they can create their own local "test" release of the DØLibrary, based on a particular offical release, but with the ability to override particular package versions. Only the package that they will be modifying needs to be physically present in their area. Any other packages that depend on their code is present via links into the distribution area. Yet all code that can be effected by changes to their package(s) can and will be recompiled as needed.

The release procedures are based on GNU make and use many other GNU products, including their version of CPP as a preprocessor for any text based file and, by default G++, their C++ compiler. The BaBar procedures are a suite of relatively simple Bourne Shell scripts and GNUmakefile fragments which a user, usually the package czar, would include in his/her GNUmakefiles. The included fragments (will) contain all the rules and targets needed to build the binaries during a release. All the user needs to do is define some specific GMAKE variables to tell GMAKE what needs to be done to the code in each subdirectory. The supplied GNUmakefile fragments fall into several classes:

- GNUmakefile.main (copied and used only by the release procedures) walks through the directory tree invoking the others.
- GNUmakefile.example is a skeleton of the makefile which must be present in *every* package directory, including the top one, that either requires some action (compilation, linking, copying of files to a release subdirectory etc.) or contains subdirectories that require action.
- standard.mk: must be included by *all* user's GNUmakefiles, usually as the last line of the file. It contains most of the rules and targets for gmake.
- arch_spec.mk: contains architecture specific declarations and rules, included by standard.mk
- arch_spec_xxx.mk: product specific makefile fragments. Contain product and architecture specific declarations and rules. To be included in user's GNUmakefile before standard.mk if the actions need the product xxx. Examples of xxx are X11, Tcl, XDR, Inventor etc.

UPS/UPD are most familiar as the "setup $< package >$ [$< version >$]" that we all use at Fermilab. They are also used to distribute all of the FNAL maintained products.

The major unresolved questions for the Joint Code Management Group center around UPS/UPD, in particular how to make it more "user friendly" and automatic at the remote end and how to make the initial "bootstrap" of FermiTools easier. BaBar uses a different and much simpler method of distribution. However, UPS/UPD has some major advantages that we are loath to abandon.

At the moment there is one person from DØ working part time on developing and understanding this system. CDF has significantly more FTE effort assigned, and is therefore driving the direction of development. DØ needs one person working close to full time for the next couple of months until we get a fully working system.

2.9 Software Development Tools

It is generally recognized that object oriented software development benefits greatly from the use of standard OO design methodolgies (Booch, OMT, etc.). Furthermore, the application of these methodologies benefits from the use of software design tools (CASE tools). The software tools working group recommended that DØ acquire a design tool for object oriented design with the following capabilities [15]:

- Supports a standard object oriented design methodology (generates diagrams, etc.).

- Automatic code generation from design diagrams.
- Reverse engineering (generates diagrams from code).

The current situation in DØ is that many people have taken a laboratory sponsored object oriented analysis and design course based on the OMT design methodology and notation. The OMT design methodology and OMT style diagrams are therefore a de facto standard in DØ and in the laboratory. DØ has acquired on a trial basis a CASE tool called Graphical Designer by Advanced Software Technologies. Graphical Designer is available for use by all DØ collaborators. The final choice of a CASE tool for use by DØ and the rest of the laboratory has not yet been made.

Software development tools are being evaluated by a joint CDF/DØ/Computing Division group called the "Methodology and Tools Working Group" [16]. This group is studying a variety of software development tools, including CASE tools, code analyzers, debugging tools, and integrated development environments.

2.10 Document and News Management

DØ currently provides two standard ways for DØ collaborators to communicate with the rest of the collaboration. The VMS VAXNEWS program (also called D0NEWS) is used for lightweight ascii news. DØ Notes are used for hardcopy distribution of arbitrary documents, including non-electronic ones such as handwritten transparencies. The WWW has also been increasingly used in recent years for distribution of various kinds of electronic documents.

It is generally recognized that the WWW is a good medium for distribution of documents within the collaboration. However, WWW based document and news serving has not eliminated the need for either VAXNEWS or DØ Notes. Some problems with pure WWW based document serving are as follows.

- Current WWW browsers lack a suitable news reading interface (no notification).
- No archival mechanism.
- Relies on unstable physical addressing (url's) instead of logical addressing (DØ Note numbers).
- Non-electronic documents.
- The need for authors to prepare multiple document formats (e.g. HTML for browsing/searching and postscript for printing).
- Labor intensive web maintanence.

DØ will definitely need to replace D0NEWS because of the phasing out of VMS at Fermilab. While, some DØ Notes currently find their way onto the WWW, the DØ Note operation has not so far been very effectively integrated with the WWW.

The DØ Software Tools Group has considered the future document and news management needs of the DØ collaboration [17] and produced a set of recommendations. These recommendations are mostly in the form of desirable features.

- All DØ documents and news should be distributed via the WWW.
- Automatic submission of documents via the WWW.

- Automatic conversion of document sources to multiple viewable formats.
- Text documents should be searchable.
- Archival mechanism.
- Notification and expiration mechanism (news).
- Embedded links using logical addresses (how?).

So far, little progress has been made toward providing these features.

2.11 Summary: Milestones and Manpower

The Software Tools group has identified needs for DØ support for

- C++ language support: jointly with CD/CDF; 2 DØ people ($<$ 1 FTE); needs to be augmented to 2 FTE.
  - Defining supported compilers and writing Fermilab/DØ C++ standards
  - Supporting a Physics Class Library to replace certain functions of CERNLIB and to provide the necessary experiment-independent basic classes.
- Object persistency support: so far, purely a DØ effort, designing our interface to the persistency mechanism and implementing DSPACK; the DSPACK group (5 people, $\approx$ 1 FTE).
- DØ program framework: the Framework group (3 people; $\approx$ 1 FTE for the next 3 months, declining to 0.5 FTE thereafter?)
- Database support: expected choice of commercial database solution in common with CD and CDF; hope to limit DØ's support requirement to 1 FTE or less.
- Script language and GUI's: provided as needed in specific applications; no FTE requirement for additional support.
- Graphics: the Graphics Working Group (3 people, $\approx$ 1 FTE).
- Configuration Management: in conjunction with CD/CDF group; DØ currently supplying 0.5 FTE which should be upgraded to at least 1 FTE for the next 3 months.

3 The Online System

The Online section of this document will be concerned with software development for the Data Acquisition, or *Online*, system of the DØ Experiment. The section will begin with a description of the architectural components of the Online system. Next we give a detailed description of each of the computing tasks which must be performed, followed by a list of the software components to accomplish these tasks. A final subsection discusses the schedule and manpower needs for the Online system.

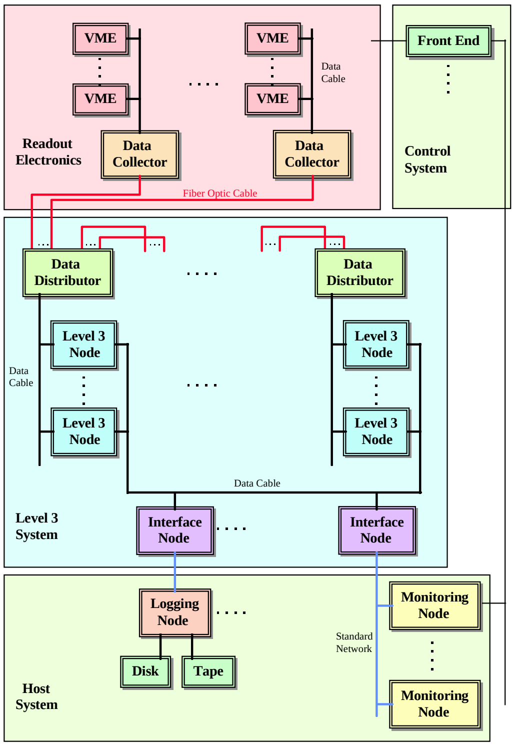


Figure 2: Online System Architecture

3.1 Online System Computing Architecture

The purpose of the Online system is to control and monitor the operation of the DØ detector, acquire data from the detector, store the data for future offline processing, and provide an interface with the accelerator systems. The principal computing components of the system are the detector *Control and Monitoring* subsystem, the *Readout* electronics with embedded processors, the *Trigger* subsystem, and the *Host* subsystem. Figure 2 illustrates the connections among these components. This subsection will give a short description of each, with an attempt, where possible, to indicate the particular computing hardware which has been chosen for the component.

3.1.1 Control and Monitoring Subsystem

Control and Monitoring (C&M) can be briefly summarized as the computing and communication tasks necessary for the operation of the detector outside of the event data path. There are C&M aspects to each of the subsystems listed below, but there are also system components dedicated solely to C&M functions. DØ will use a number of *front end* processors with various *field busses* (serial or parallel interfaces) for these functions. All of the front ends are planned to be single-board processors residing within VME crates (although it is possible to extend the model to include standalone processors). There will be two classes of front end processors in Run II — split between *new* and *old*. Much of the software effort in the C&M regime will be in providing a consistent view of their functions.

The new front ends will use VME-based Motorola 68000 or PowerPC family processors.

They will reside within the readout crates for the Silicon Tracker, Fiber Tracker, Preshower, and (possibly) Muon detectors. Since the VME backplane of these crates must be dedicated to readout operations, these front ends must acquire information via an external field bus, currently expected to be an IEEE 1394 connection to the VME Readout Boards (VRBs) in the crate. These processors will use the VXWorks operating system. The front ends will perform calibration and monitoring tasks, including alarm generation. These processors communicate with the Host system via Ethernet.

DØ will also continue to use a set of VME-based Motorola 68000 family processors communicating via a Token Ring network. These processors use the pSOS operating system, and run code developed by the Fermilab Particle Beams (formerly Accelerator) Division Controls Group. These front end systems have access to their local VME crate, adjacent VME crates via a *Vertical Interconnect* bus extender, and remote devices via a MIL-1553 field bus. There will be approximately 20 such front end processors, mostly in the Moving Counting House, in the Run II system.

The Vertical Interconnect (VI) is used to extend the addressable memory space of the processor to encompass adjacent VME crates. Actions over the VI are thus memory read or write actions. These actions are used primarily to load constants into readout boards or load code into subsidiary processors.

The 1553 field bus is used to connect to *Remote Terminals*, which in turn are connected to various electronics hardware, for example power supplies, within the detector. The 1553 also connects to a set of *Rack Monitors*, which have environmental monitoring functions — temperatures, air and water flows, voltages and current flows, and leak detection. The code in the front end processors allows remote access and control of these devices, and also compares a periodic scan of readings against nominal values for use in alarm generation.

A special set of front end processors controls the DØ High Voltage (HV) system. A standard C&M node acts as the interface to each set of HV crates, communicating via the VI with one master and several subordinate processors. The master and subordinates run code customized to the HV system, and communicate via the VME backplane with the HV modules.

The Token Ring *Gateways*, processors with both Token Ring and Ethernet connections, are important components of the C&M subsystem. The Gateways act as the interface between those two network mechanisms. Each Gateway is currently a μVaxIII processor running the ELN operating system. It is in a VME crate containing a Token Ring network board. For Run II the Gateways will be changed to VME-based Motorola processor boards, and will run the VXWorks operating system.

3.1.2 Readout Subsystem

The digitization electronics for the DØ detector reside in VME crates. In each crate is a VME Buffer Driver (VBD) through which the data path readout occurs. The VBDs are preprogrammed to gather information from the digitizing electronics within the crate, and perform a level of buffering before readout. Under the control of the trigger system, data flows from the VBDs to a *Data Collector* via a custom 32-bit wide *Data Cable*. Each of several Data Collectors is responsible for a set of VME Readout crates.

The code within the VBDs which controls the readout is considered to be specific to and within the realm of the detector hardware groups, and is not considered further within this section. Similarly, the code controlling the operation of the Data Collectors is considered to be a portion of the Level 3 system, and not generally within the confines of the overall Run-Time section. But, as with other portions of the Level 3 system, various communication paths to control and monitor Data Collector operations are later discussed.

3.1.3 Trigger Subsystem

DØ uses a multi-stage trigger mechanism consisting of 4 *Levels*. A Level 0 trigger is formed in hardware from forward scintillator counters, and essentially indicates the presence of an interaction within the detector. Level 1 is a deadtimeless fast hardware trigger, formed from pickoffs of various detector signals. Level 2 uses similar hardware signals as inputs to processors within which more sophisticated algorithms are run. When a Level 2 trigger is accepted, the readout of the complete detector is initiated. The event data passes into one of a set of Level 3 processor nodes, where nearly complete event reconstruction is performed. If the event satisfies a Level 3 trigger criterion, then the event is passed on to the Host system for logging.

Although the Level 1 trigger is determined solely in hardware, control and monitoring of the Level 1 Trigger activity is also necessary. The Level 1 trigger framework interfaces to the remainder of the Online system via the *Level 1 Trigger Control Computer*, also known as the *TCC*. In Run I the TCC was a VAX running the ELN operating system; in Run II the TCC will be a single board Power-PC processor using the Windows NT operating system. The software developed for the TCC is in the domain of the Level 1 trigger development group, so its details are not included in this document. However, the TCC does interact with many other components of the Online system, and these interactions are considered.

Similarly the Level 2 system consists of processors with software in the domain of the Level 2 trigger development groups. Level 2 will be constructed from a more diverse set of processors, including a *framework* processor similar in function to the Level 1 TCC, several detector-specific processors, and a *Global* processor. The specific hardware choices for Level 2 include single-board DEC Alpha processors and CMAPS processors. As with Level 1, the focus within this document will be on the necessary communications with the Level 2 system and will not include the internal details.

The data path through the Level 3 subsystem begins at the fore-mentioned *Data Collectors*. A set of *Data Distributors* act to direct an entire event via a Data Cable to an available *Filter Node* wherein a trigger decision is made in software. Accepted events are passed, again via a Data Cable, to an *Interface Node* and then on to the Host. The Interface Node acts to move the event from the Data Cable path to a standard network path.

Additional Level 3 components coordinate the activities of the Data Collectors, Data Distributors, and Filter Nodes. The external C&M interface to the Level 3 subsystem will be via these coordinating components. The Level 3 Filter Nodes and these additional Level 3 components will be PCs running the Windows NT operating system.

There are two significant software components to the Trigger subsystem — the C&M aspects and the actual filter algorithms. The former will be discussed in following sections. The filter code is not discussed in this Run-Time section; Level 1 and 2 codes are considered to be set by the detector hardware groups, and Level 3 filter codes are in the realm of the Algorithms group.

3.1.4 Host Subsystem

The Host subsystem is the center of operations for the DØ detector and data acquisition (DAQ) system. The Host contains the processors and peripherals associated with the functions of run and configuration control, detector control and monitoring, data logging, data monitoring, and accelerator monitoring. Most of these functions will be accomplished in dedicated or semi-dedicated processors. The principal Host nodes will run the UNIX operating system, but the specific hardware has not yet been determined. It is also expected that there will be some component of Windows NT PCs within the Host subsystem.

Operator control of the detector and DAQ system will be by interacting with a set of applications residing on Host nodes. These applications are likely to be distributed among several processors, as appropriate for their specific functionality. The core applications will maintain knowledge of the configuration of the detector, the configuration of the trigger, and the state of the DAQ system. These applications will also be the center of control of these configurations and the DAQ system.

A set of *Logging Nodes* will perform the event data path management functions. The number of Logging Nodes (and associated Level 3 Interface Nodes) can be adjusted to match both the required event data bandwidth and the needs for reliability via redundancy. Each Logging Node will buffer event data to local disks for later spooling to either local or remote (in the Feynman Computing Center) mass storage devices.

In a manner similar to the Logging Nodes, one or more *Event Monitoring Nodes* can also receive event data from Level 3. The Event Monitoring Node(s) will analyze, or distribute via the network to other Event Monitoring Nodes, selected (by trigger type) events. The monitoring functions include the accumulation of histograms useful in determining the operational state of the detector, the trigger system, the filter and reconstruction codes, and the accelerator.

Although many detector calibration functions will occur in either front end processors or Level 3 processors, it is also possible that some calibration constants will be determined from data at the Host level. The Monitoring Nodes will also serve this function, either as a by-product of normal event monitoring or as a dedicated operation.

Additional monitoring functions will be performed via the controls path instead of the event data path. Applications running on Monitoring Nodes will have access to detector information via the Controls and Monitoring path previously described. One particular set of monitoring operations will comprise the *Significant Event*, or *Alarms* system.

The Host system also serves as the interface to the Accelerator computing system. Information can be exchanged between the Accelerator and DØ via a standard network link.

3.2 Online System Tasks

In this subsection we will list the various and numerous computing tasks needed for the Run-Time system. The tasks will be roughly grouped by functionality, though there are clearly tasks which fit several categories.

It is expected that software will need to be written or obtained for each of these tasks. However, there is not a simple one-to-one relationship between the tasks to be listed and the packages or applications to be written. Often several distinct functions can be easily gathered into a single application. Conversely, often several coherent applications are needed to fulfill a single function. Hence this subsection will concentrate on delineating the tasks, with the following *Software Components* subsection addressing the array of tools and applications expected to fulfill the needs.

As indicated in the *Architecture* subsection, this document does not address the specific processing tasks related to readout of the electronics, the Level 1 and Level 2 triggers, and the internal Level 3 data flow management. For these details the reader is referred to the appropriate sections of the DØ Upgrade Technical Design Report. This subsection also does not address the software needs related to the Level 3 Filter code, nor to software necessary to simulate the operations of the Trigger subsystem. These issues are discussed elsewhere in this document.

The task categories listed below roughly parallel the Online subsystem architecture as described in the previous subsection. There are software tasks associated with each of the major components and with the communication between the components.

- *Downloading*

  The Run-Time system contains numerous electronic components which need to be initialized or otherwise have their state modified. In addition, many embedded processors must receive their code from an external source. In general, a scheme is needed to store and download from a persistent source, presumably on the Host, the following information:

  - code in embedded processors dedicated to detector readout;
  - code and parameters in embedded processors dedicated to trigger processing at the detector level;
  - code and parameters in Level 1, Level 2, and Level 3 trigger processors;
  - code in embedded processors dedicated to detector monitoring and calibration;
  - constants, e.g. pedestals and gains, to digitization and readout electronics; and
  - constants, e.g. nominals and tolerances, to front end processors engaged in detector monitoring.

- *DAQ Control*

  The DAQ system is presumably under operator control. An operator interface is thus required, plus a suite of applications to interpret, expand, and execute the operator instructions. The range of activities under direct or indirect operator control include:

  - detector configuration, e.g. assignment of detector resources to specific users or purposes;
  - trigger selection, specifically the activities of the Levels 1, 2, and 3 of the Trigger subsystem;
  - run control, including beginning, pausing, resuming, and ending runs;
  - downloading activities, as listed above; and
  - data disposition — discarded, temporarily retained, or permanently recorded.

- *Data Path*

  Event gathering, routing, and disposition is all under computer control. The task list thus includes those processes responsible for moving the event data through the Level 3 and Host systems to final persistent storage. These processes must also provide control and monitoring points. It is desirable that their operation be flexible so as to differentially route and dispose of events from simultaneous runs or from a diverse set of triggers.

- *Calibration*

  Whether detector calibration occurs in front ends, Level 3 processors, or in the Host, the software to determine the constants and manage their storage must be provided. It is expected that much of the framework of the calibration processes can be shared, but there will remain numerous detector-specific components.

- *Detector Monitoring*

  A range of tasks must be in place to have a complete detector monitoring scheme. The hardware must be accessed at the front ends, which in turn should perform alarm scanning. The alarms should proceed to the Host where they, in addition to alarms generated elsewhere in the system, are collected and distributed to operator displays.

The front ends also must provide the mechanism by which the detector hardware can be queried, either as part of a specific investigation or part of a periodic logging of detector conditions. Associated Host level tasks include a *Parameter Page* and a *Logger*.

Software is required for the High Voltage system, both as an interface to the HV hardware and as an interface to the operator.

- *Event Monitoring*

  There are two aspects of Event Monitoring — one for analysis of the performance of detailed detector elements and one for broader analysis of the performance of the detector and trigger system as a whole. This monitoring is to be accomplished by an examination of events in the data stream. The event information needs to be distributed to subsidiary processes, which can share a common framework in which to do the analyses.

3.3 Online Software Components

Software development for the Run-Time system consists of making a coherent plan for providing the tools and applications to fulfill the tasks outlined above. The Run II Online system will be built upon the lessons and expertise of Run I, with many of the same components. However, the demands of higher performance and new detector elements coupled with the desire to join the greater DØ computing migration to UNIX, NT, and C++, requires most existing applications to be replaced and many new applications to be created.

The previous subsection, on *Online System Tasks*, listed the jobs which have to be performed by the Run-Time software. This subsection gives our plan for the principal tools and applications to address those tasks.

3.3.1 Interprocess Communications

The design of the Run-Time system with many distributed applications and processors requires the existence of a robust and efficient interprocess communication tool. In fact, most of the operations of the Online system consist of nothing more than the exchange of messages between tasks and the simple actions invoked by these messages. Since this is a pervasive activity, it seems sensible to have a uniform solution which can be shared by system components.

In Run I the interprocess communication was based upon DECNET (for remote processes) or VMS Mailboxes (for local within-processor processes). Many applications further relied on the ITC product, a layer above the low-level communication calls. ITC provided a simple client/server mechanism with named targets, message buffering, and asynchronous program flow. However, ITC relied on the calling application to do most error handling and recovery. Another package, CLSPKG, provided a layer above ITC in which much of the error handling was automatic. In the end the Run I system was an amalgam of many messaging schemes. The robustness of several was questionable, and led to numerous problems. It is our intention in Run II to choose an effective communication method and use it wherever possible.

In the spirit of the move to more modern software paradigms, we have considered the use of CORBA-compliant Object Brokers as a solution to the interprocess communications need. Several commercial products were marginally considered, but we felt the need to reduce costs dictated a search for public domain products. Two such products, ILU and ACE, though not strictly CORBA-compliant were considered for possible use. In the end we have decided to forego the complexity of the Object Broker approach and rely more on the simplicity of direct messaging.

Our chosen solution is a modernization of the current CLSPKG client/server package. The lowest internal layer will be a new product, D0IP, which handles communication via IP sockets. The new package, to be constructed as a messaging class, will give the features of connection- or connectionless-based messaging, name serving, error correcting, and internal monitoring. Because the Run-Time system is potentially constructed from processor elements with differing internal architectures, the messaging class will also need to handle record formatting and byte swapping. This new client/server package is currently in the design phase.

3.3.2 Control and Monitoring

Control and Monitoring functions will require the largest number of specific applications, ranging from Front End applications to assorted Host level applications. DØ is currently evaluating, and expects to use, many components of the EPICS control system to satisfy many of these needs. EPICS, which has matured considerably since the Run I system was constructed, potentially provides not only a framework in which to build customized applications but also a number of specific applications which can be used off—the—shelf. Several of the applications listed in this section are being developed in conjunction with the EPICS evaluation, and several others will be designed and constructed to operate within the EPICS framework.

A list of the principal C&M applications follows:

- *Hardware Database*

  A principal component of a Control system is a database wherein reside the details of how a hardware device is accessed, the format of its data, data transformations from raw to engineering units, possible alarm limits and actions, and many more associated constants. In Run I DØ used an RDB database to store this information. In Run II there will be a similar need, but the specific database to be used is still under study. Note also that EPICS requires the presence of a database, so that the Run II database must additionally serve EPICS needs.

  A database must come with a set of built-in and DØ-specific applications to access it:

  - *Data Entry Program*: An interactive and/or batch application which enables entry of information into the database.
  - *Database Monitoring Tools*: Typically a part of a commercial database, these tools allow monitoring of database activity, provide for making backups, and aid in error recovery.
  - *Download Application*: As will be described later, a means to retrieve database information to load distributed local (front end) data stores is required.
  - *Database Server*: In Run I database access performance by remote processes was greatly improved by creating a server process with specially tuned privileges. The server was accessed from remote clients by an interchange of network messages. Depending upon performance needs, a similar client/server arrangement may also be necessary in Run II.

- *Old Front Ends*

  A number of VME-based 68000 processors acting as the front ends for the Control system are to be retained from Run I. These processors use the pSOS operating system, and run code developed by the Fermilab Beams Division Controls Group. Prior to Run II DØ will need to bring its nodes up to the current version of the front end code. Additionally, DØ has the need for block transfer mode over the 1553 bus, so some modification of the 1553 driver will be required.

Although the code in these front ends is similar for all nodes, the operation of each is customized by a set of memory resident tables. The customizations allow for different hardware configurations. Some mechanism is necessary for storage of these custom tables in addition to storage of the standard tables and code. In Run I this requirement was satisfied by a process running on a SUN workstation which used the Token Ring network link to retrieve or load blocks of memory, including the program code itself. The Beams Division is porting this code to a NT platform, and DØ will use this new application for this function.

A large number of parameters must also be loaded into each front end to control operations specific to each monitored hardware device. For example, the local alarm scan requires a nominal value and tolerance be loaded in a local table. This information ultimately resides in the Hardware Database, but must be downloaded to the front end. The application which performs this function will be described later.

- *New Front Ends*

  The front ends associated with the Silicon detector, Fiber Tracker, and Preshower detectors will need to run code similar to the Old front ends. Since these nodes have no legacy functionality to maintain, we expect to configure them as EPICS front ends, and hence utilize the EPICS code. The specialized needs of DØ will be added as EPICS functions — access of data over the 1394 bus, accumulation of local detector information and report of such information, alarm scanning and generation, and calibration activities.

  The integration of the Old and New front ends into a uniform system will be complex. We expect that the EPICS system will perform as the backbone, with other systems integrated by surrounding them with a translating interface layer. Thus, the Old front ends will be accessed by EPICS calls which get massaged into something understood by the older systems. (This action will be performed by the Gateway code, to be described later in this subsection.)

- *Front End Management*

  As indicated earlier, the front ends maintain a number of parameters specific to the hardware devices which they control or monitor. Rather than store this information independently within each front end, the data is collected and stored within the Hardware Database on the Host. An application is needed to easily compare the contents of Host and Front End databases and to further move the information as necessary. In Run I this application was based upon the RDB database, the CDAQ control package, and the Old Front Ends. The necesary Run II application will instead use a new database, the EPICS control package, and a combination of Old and New Front Ends.

- *Gateway*

  The Gateway node, which acts as an interface between Token Ring and Ethernet networks, must also act as an interface between the EPICS control system and the Old Front Ends. The existing Gateway code must then be modified to understand the message protocol employed by EPICS and to translate such to or from the ACNET protocol understood by the Front Ends.

- *High Voltage*

The High Voltage system contains both specialize front end processors and Host applications. We will retain the front end code used for Run I, though some modifications might be necessary to account for new detector needs. The Host level applications need considerable change to adapt to the new operating system environment(s). The principal Host application is an interactive HV control and monitoring program, through which operators may control the HV system. Following the Run I example, another application will monitor the state of the HV system to provide a control room display. Finally, the HV system will need to be linked to run and configuration control operations.

- *Cryogenics and Gas Systems*

  In Run I the detector Cryogenics and Gas systems were controlled and monitored via tasks running with front end processors. Similar activities and tasks will be necessary in Run II, and some amount of code updating and development will be necessary.

- *Parameter Page*

  Generalized detector C&M is achieved with the use of a Parameter Page, through which hardware devices can be operated. In Run I the parameter page was a custom CDAQ-based application. In Run II we are expecting this functionality to be part of the EPICS suite of applications.

- *Clock Server*

  The DØ Clock is a device connected to the Accelerator by which precise events in the accelerator cycle can be located. The Clock itself is controlled as many other devices through an Old Front End processor node. Since the Clock can maintain a configuration state, plus can be programmed to provide multiple possible signals, a Host process exists to manage its operation. This process is called the Clock Server, because all possible Clock operations are managed through it. The Run I Clock Server will be modified to fit the Run II Host environment.

- *Alarm System*

  The Alarm System is based around an Alarm Server, a process to which all *significant events* are sent for distribution to multiple clients. The Alarm Server maintains the current state of the experiment in terms of detector hardware and DAQ software status. As already mentioned, one function of the Front Ends is to generate alarm messages when a device is noted to be out of tolerance. Similarly, software processes can generate significant events to note changes in state of the experiment or noteworthy occurrences. There are multiple consumers of this information — Alarm Displays, an Alarm Logger, and a run control monitor — which are connected to the Alarm Server. The Run II system will closely follow the architecture of the existing Run I Alarm System, but all tasks will need to be updated to match the Run II computing environment.

- *Detector Data Logger*

  Much potentially useful information on the operational condition of the detector remains outside of the scope of the normal event data. For example, environmental conditions can affect the calibration and hence analysis of detector operation. To aid in the eventual reconstruction effort, it is useful to record many detector parameters. A Data Logger (distinct from the Event Data Logger) application on the Host acquires this information and stores it, presumably in an easily accessible database format. A Run II version of existing code will need to be created.

3.3.3 Configuration and Run Control

In Run I the *COOR* application was the center of operation of the DAQ system. For Run II we expect a similar arrangement, with an updated version of COOR serving many functions: as the repository of the current configuration of the detector, the clearinghouse for requesting configuration changes, the repository for the current state of DAQ runs, and the process which ultimately controls these runs. COOR operates with a set of companion programs which will also have Run II updates:

- *COMM_TKR*: COOR uses COMM_TKR as its instrument for using the Controls system to download information into the detector. The Run II COMM_TKR will be build upon EPICS.
- *TAKER*: TAKER provides the operator interface to COOR, instructing trigger or configuration changes and directing the sequence of runs.

Note that for the Test Beam operation in 1997 that the complete COOR system will not be used. Instead the DAQ system will be comprised of components of the Fermilab Computing Division DART product. At the Test Beam, DART will be interfaced with the New Front Ends in a manner such as to use these processors in a dual mode, both for the traditional control and monitoring and for the acquisition of event data.

3.3.4 Data Path Operation and Monitoring

The Run I application *Data Logger* was responsible for receiving event data from Level 2 and spooling it to disk. A similar function is necessary in Run II, as one or more Host processes receive data from the Level 3 Interface Nodes and spool the events. In Run I, associated processes managed the transfer of event data to tape. In Run II we expect that event data will be shipped to the Feynman Center for writing to robotic mass storage. This activity will be under the control of the Computing Division, and DØ will request their assistance in providing a reliable final stage in the data path.

Monitoring of the data path is critical to operation of the experiment. Each component of the trigger and the Data Logger will provide monitoring functions. The information will be used to drive displays, produce alarms, and generate performance statistics.

3.3.5 Calibration

A number of applications related to the calibration of the detector are necessary:

- *Front End Calibration*: The new detectors which employ a VME Readout Board (VRB) will be equipped with a New Front End processor in which will reside the software for calibrating the local electronics. We expect to make this functionality part of the overall EPICS front end structure. Once calibration within the local processor is complete, then the resultant information will need to be transferred to the Host for storage. This will be achieved by a Control path operation.
- *Level 3 Calibration*: Detectors without a New Front End, notably the Calorimeter, will perform calibration within Level 3. Pedestal and pulser events will be transferred to Level 3 filter nodes for analysis, with the final results being transferred (via the common Client/Server messaging) back to the Host for storage.

- *Host Calibration Tasks*: The operator interface to calibration activities in either the Front Ends or in Level 3 will be on the Host. Additionally, all calibration results eventually return to the Host for storage. In Run I this information was stored in DBL3 databases; in Run II we expect to store information in a database of form yet to be determined.

3.3.6 Event Monitoring

In Run I a subsidiary process of the event Data Logger selected events from the data stream and distributed them to client processes. In Run II DØ will attempt to use applications developed as part of DART for the distribution of events to consumers. The consumers, called EXAMINEs in Run I, will be built upon the standard DØ package framework. There will be several such applications, for the analysis of detector specific information and for the analysis of global experiment operations.

3.3.7 Accelerator Interface

A large amount of information is available from the accelerator during the operation of the experiment. Some small amount of information makes it into the event data stream, mostly via Level 1 trigger scalers. Other information must be recorded outside of the event data stream. We expect to produce an application to acquire the accelerator data and to log it to a Host database. This information will be used in the determination of livetime and luminosity.

It is also necessary to feed back information from DØ to the accelerator. In Run II the most important such information will be on the position of the beam within the detector. Misalignment of the beam will severely affect the ability to locate displaced vertices, so it is advantageous to provide immediate feedback. This will most likely be achieved within an event consumer application which determines the event vertices and then, by network connection, transfers beam information back to the accelerator system. The modes of communication with the accelerator are yet to be determined, so this application has yet to be designed.

3.3.8 System Performance Monitoring

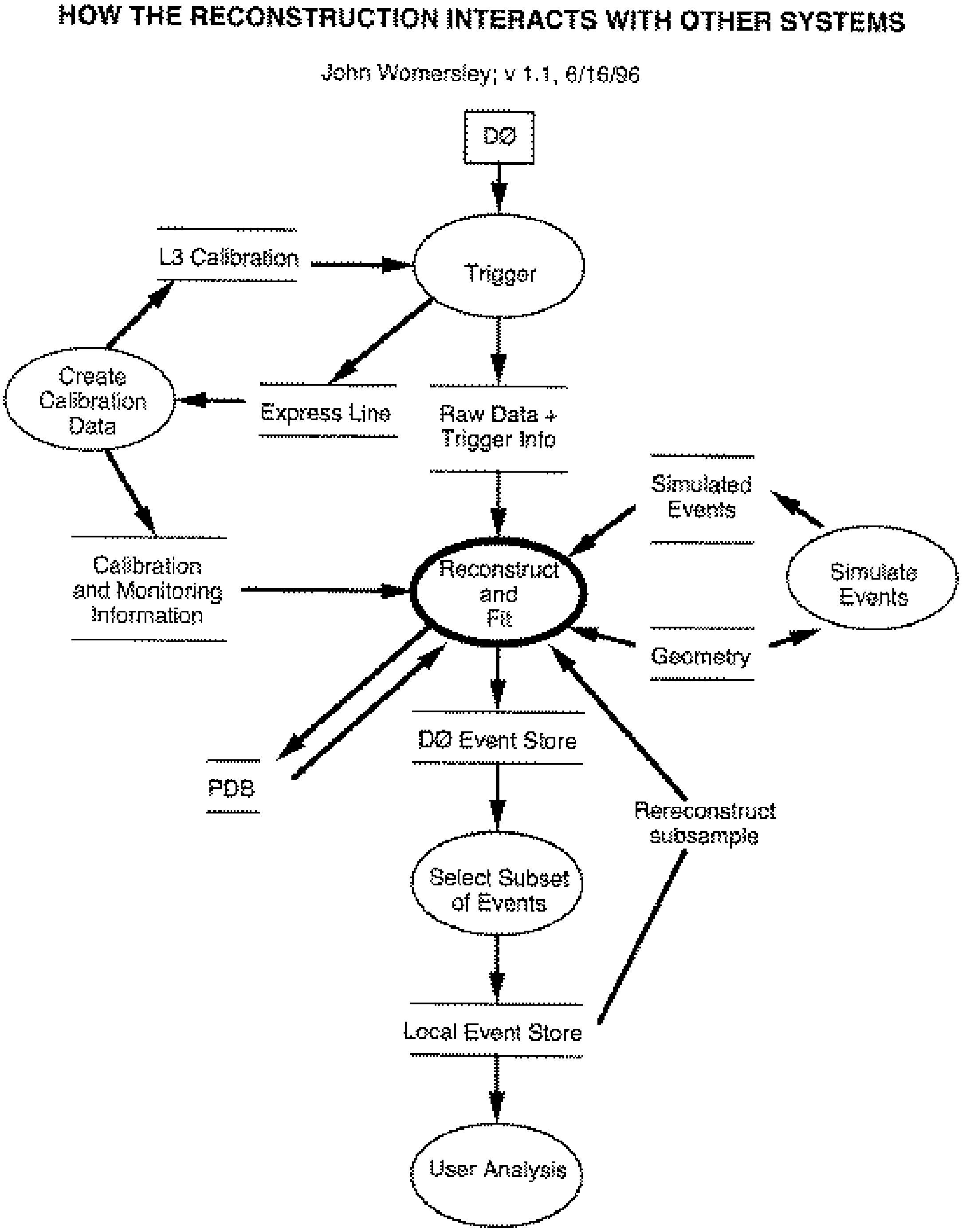
It is important that the Online system be reliable and efficient. To that end, the operation of the computing system must be closely monitored to determine processor, network, IO utilizations. In addition, anomalous performance must be understood to aid in failure mode analysis and recovery. It will be important to provide displays of system performance and to keep a concise log of system activity.

3.4 Schedule and Manpower Assessment

A Table or Figure summarizing the milestones contained within the larger MS Project schedule. Manpower profiles can also be extracted from this schedule.

4 The Reconstruction Code

The reconstruction system must take raw data from the detector and process it to yield the measured physics objects (such as electrons, jets, missing transverse energy) required for physics analysis. The code required for this is the responsibility of the Run II Algorithms group, led by Serban Protopopescu and John Womersley. This group is also responsible for the code used in


HOW THE RECONSTRUCTION INTERACTS WITH OTHER SYSTEMS

John Womersley; v 1.1, 6/16/96

Figure 3: How the reconstruction system interacts with other processes.

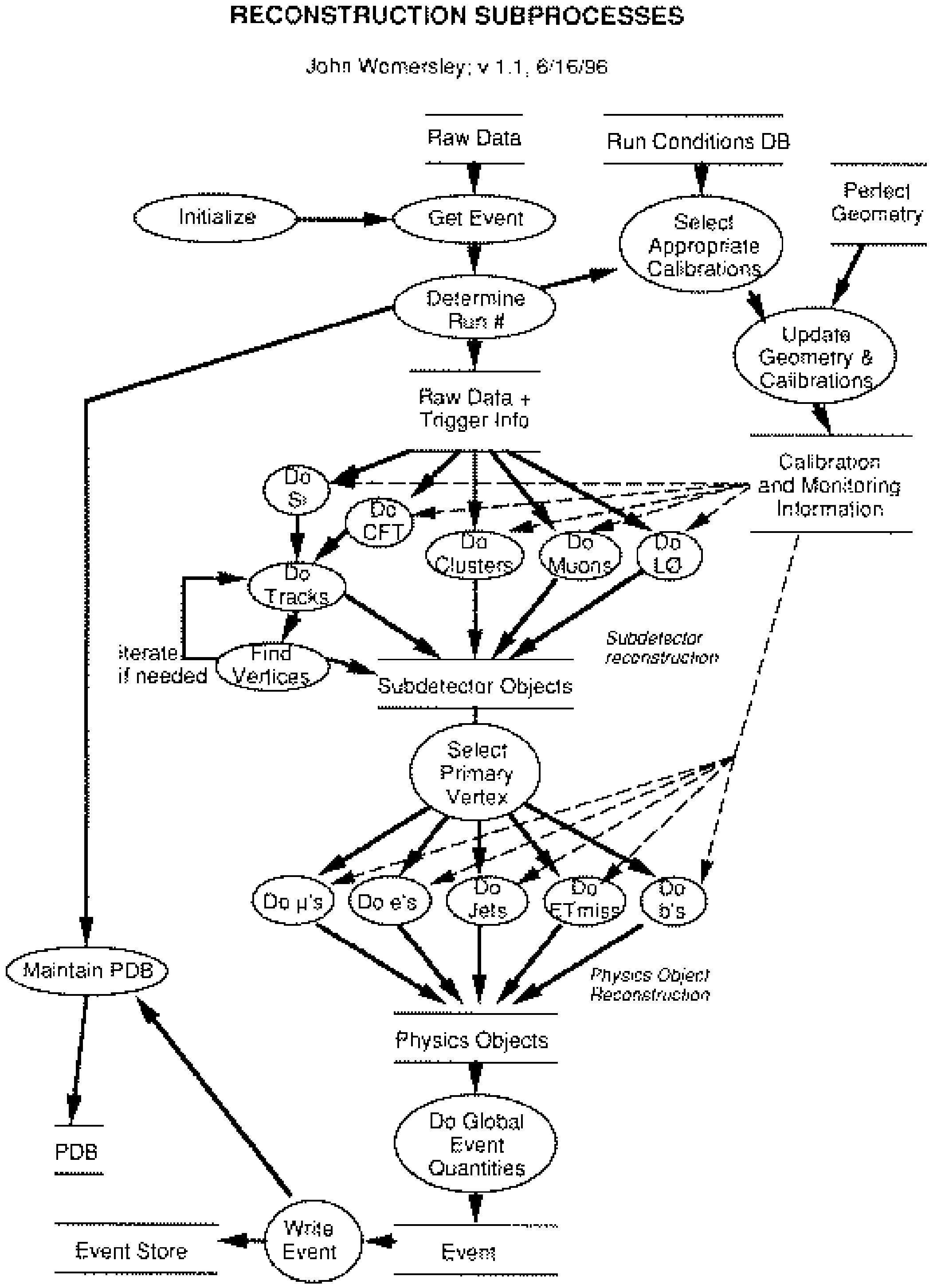


Figure 4: Reconstruction subprocesses.

the software (level 3) trigger system and for Monte Carlo programs to simulate the detector's response.

The interactions between the reconstruction system and other software systems are summarized in Fig. 3. As indicated, raw data is received from the data acquisition system and processed through the reconstruction to an event store. Simulated data should also be able to pass through the system. Information on the detector geometry, and on calibrations, monitoring and alignment, will also be required.

We have attempted to break the task of reconstruction down into smaller units, as shown in Fig. 4. As far as possible, reconstruction should be factorized into:

1. Subdetector reconstruction (e.g. clustering in the calorimeter, finding tracks and vertices in the tracker)
2. Physics Object Reconstruction (e.g. associating tracks and clusters to form electrons, associating tracks to form K^0's, etc.)
3. If necessary, subsequent association of physics objects at a higher level (e.g. an electron and missing transverse energy to form a W candidate).

The present activities of the Algorithms group concentrate on the first task listed above: reconstruction of subdetector objects. We have active subgroups working on Silicon Tracker reconstruction, Fiber Tracker reconstruction, Calorimeter reconstruction, Muon System reconstruction, Alignment and Calibration, Level 3 Algorithms, and Monte Carlo simulations.

The tasks of the subgroups are:

- **Silicon Group**: create silicon hits including clustering (includes dealing with calibration, geometry: assign a person to the calibration and alignment group); reconstruct track segments in silicon (note that this is the full tracking for $|\eta| > 2.5$); find the z-location of the proton-antiproton vertex (vertices); find displaced vertices.
- **Fiber Tracker Group**: create fibre hits including clustering (includes dealing with calibration, geometry: assign a person to the calibration and alignment group); reconstruct track segments in fibers and overall central tracking (fibres+silicon).
- **Calorimeter Group**: create calorimeter hits (includes dealing with calibration, geometry: assign a person to the calibration and alignment group); create clusters of energy (given a vertex position); include the ICD and the central and forward preshowers.
- **Muon Group**: create muon hits (includes dealing with calibration, geometry: assign a person to the calibration and alignment group); reconstruct track segments in the muon chambers.
- **Calibration-Alignment Group**: determine what information is required, and on what timescale (slow vs. fast?); determine what special programs will be needed, what special triggers, and what dedicated data processing; develop a geometry database, allowing for displacements, misalignments, time-dependence, etc.)
- **Level 3 Algorithms Group**: provide efficient algorithms for the Level 3 trigger (which may or may not be the same as offline algorithms); provide the tools for simulation and verification; provide production releases and code reviews
- **Monte Carlo Group**: maintain a realistic, working simulation of the detector at all times. The geometry should evolve to a common solution with the reconstruction processes.

We are postponing development of global reconstruction and physics objects in accordance with our schedule (see later); there will be a shift in emphasis from subdetector to global reconstruction in roughly mid-1997.

4.1 Status and Plans of the Subgroups

The responsibilities of the fiber tracking group encompass not only the reconstruction of hits and tracks in the central scintillating fiber tracking system, but also that of global tracking. As the needs of the two charges are quite different they will be treated separately in the following two subsections. Clearly, these reconstruction tasks depend on other groups' efforts, and provide information needed by subsequent analysis packages. It should be noted that FORTRAN–based tracking reconstruction packages (TRF and BETA_TRACK) exist and have been used for many analyses in the past. There is also an effort underway to find tracks using a conformal mapping technique which was developed for the STAR collaboration (see D0Note 2893). This document does not address those efforts, restricting itself to the object–oriented, C++–based implementation now envisioned for the Run II reconstruction.

4.1.1 Scintillating Fiber Reconstruction

Raw Data

The scintillating fiber detector reconstruction uses as input the raw data composed of fiber number and pulse height. Monte Carlo simulation of events and the detector's response is an essential part of any reconstruction effort. To develop reconstruction code, simulated data which mimics that expected from the detector will be used. A large body of data exists from test beam and cosmic ray runs which characterizes the intrinsic response of the detector (see references), and it is envisioned that this information will be used to improve the simulation of the detector response. The scintillating fiber detector geometry is currently implemented in `UPG_GEANT` as thin cylinders of plastic. The fiber doublets are not explicitly modeled, rather, the digitization arising from discrete fibers is handled in software. This allows the detector description to be quite flexible. Indeed, there currently exist two detector geometry proposals, differing in the radial placement of the stereo layers. The bulk of the current reconstruction effort is being devoted to a study to determine whether the design favored by hardware considerations can indeed be used for physics analysis. Clearly, there is a dependence of the fiber tracker simulation on the details of the silicon vertex geometry, and conversely, the calorimeter and preshower simulations are affected by the exact implementation of the fiber and support system geometry. Currently, aside from the carbon fiber cylinders, no other support or readout material is in place. Close cooperation with the group responsible for Monte Carlo simulations is envisioned.

There is often a need for very fast simulation of the detector's response to physics events. This need often arises when developing triggering algorithms, where very large rejection factors are required. It is expected that this need will be fulfilled by the Fermilab product `MCFAST`. There is still need for more extensive verification of the scintillating fiber implementation within this framework.

Clustering and Hit Finding

The raw data will be clustered if neighboring fiber channels are hit. The clustering algorithm has been extensively investigated using cosmic ray data and will be adopted into the offline reconstruction. The fiber cluster will then be tranformed into the appropriate measurement point by accessing the detector geometry database. The difficulties inherent in this transformation are directly coupled to how well the detector is built, assembled and installed. Accurately placed fibers mounted on rigid cylinders coaxial with the beam will greatly reduce the amount of work required of the alignment and calibration group. Liaisons with both the hardware group responsible for constructing the detector and with the aforementioned alignment and calibration group have been established and intimate contact will be maintained. Furthermore, attention is being given to realtime online monitoring, to provide the level 3 triggering framework not only its requisite precision, but also updated lists of missing or dead channels. Schemes for handling such missing fibers are also curently being investigated.

4.1.2 Fiber Tracking Reconstruction

The problem of pattern recognition in the fiber tracker will be attacked using an iterative road–following method employing the Kalman filter algorithm to reconcile the track projection and the measurement. Details of this philosophy can be found in D0Note 2958. The object–oriented–analysis paradigm being adopted for the Run II reconstruction is characterized by the need for an extensive initial design phase. As the design and code implementation are closely coupled, the process is very iterative; a good design can lead to a good program, but the deficiencies in the design are often not uncovered until the coding process is underway.

TRF++

The principal task of finding and fitting tracks using the scintillating fiber hits takes place within the TRF++ framework (see http://www.bonner.rice.edu/adams/trf++). The initial object–oriented design, including class and state diagrams as well as event traces, is largely complete. The first C++ implementation of this design is also in place and is currently being ported to different platforms and being tested with various compilers. The program is also being used for coding and algorithm optimization and design review and modification.

The first version of TRF++ is limited to track reconstruction in the scintillating fiber detector with the separated stereo geometry "XU–XV". It is currently a standalone program with internal event and detector simulation. Particles are generated with momenta flat in $1/p_T$ with $p_T > 150\ MeV$ and with a density of $dN/d\eta = 10$ to approximate the hit density in real events. The tracks are generated with a gaussian vertex z distribution ($\mu = 0, \sigma = 25cm$) and required to cross the outer fiber barrel. Hits are obtained by smearing the track intersection with the fiber cylinders by $100\mu m$ in the appropriate dimension. This simplified simulation clearly neglects the effects of correlated particle production (jets), multiple scattering, nonuniform magnetic field, production of secondary particles, and noise. The simulations are performed with varying numbers of interactions (1–8) per event, and full track reconstruction is performed independent of the Monte Carlo information. The figures of merit for the Monte Carlo and reconstruction comparisons are very good (high efficiency (99.6–99.9% for $p_T > 150\ MeV$) and low numbers of fake tracks ($\sim 0.1\%$). The reconstruction times are currently within the expected budget (~ 5 seconds for 3 interactions). Efficiencies and timing results, however, represent a snapshot of current activities and will change. Reconstruction times will rise as the simulation complexity increases, but it is hoped that the reconstruction can be speeded up by optimization of the code and algorithms.

4.1.3 Global Tracking Reconstruction

The definition of global tracking in the upgraded D0 detector has not yet been precisely defined. However, at the very least it encompasses the connection of the central silicon and fiber detectors into a central track, and the ability to associate such tracks with calorimeter objects (isolated electromagnetic showers, jets, or even individual cells or towers) and tracks reconstructed in the outer muon system. It is intended that the task of global tracking will be accomodated within the TRF++ framework. Details of its implementation, however, remain to be decided. Although it is fairly clear that the problem of associating tracks reconstructed in the muon system with those in the central detector will be solved by merging track vectors, it is not clear that a similar strategy will be adopted to combine the scintillating fiber and silicon detectors. In the central region, at least, tracks found in the fiber detector will be propagated into the silicon analogously to the current scheme employed within the fiber detector. Tracking in the forward region, especially that in the transition region between fiber and forward silicon disks, will be especially challenging. In principle, the preshower detectors can be included as an additional measurement, but whether this occurs as a part of the tracking reconstruction or as a part of the later global analysis remains to be determined.

Interactions With Other Groups

1. *Framework.* The current TRF++ is written as a stand–alone package incorporating both simplified event and detector simulation to generate the raw hits used in the tracking reconstruction. An easy–to–use framework to solve the event I/O problem and provide a framework for interaction with the rest of the reconstruction program is needed.

2. *Monte Carlo.* It is essential that a well–coordinated, well–documented and easy–to–use Monte Carlo simulation which accurately reflects the real detector be maintained. The ability to manipulate simulated events after the time-consuming GEANT processing is completed is essential (i.e. changing efficiencies, adding noise, combining events, distorting geometries).

3. *Event Display.* The task of tuning and debugging the reconstruction will be greatly eased with a useful event display. The utility of such a tool can not be overstated.

4. *Level 3.* Having the reconstruction code run as part of the online filter (albeit with different run–time options) would greatly simplify the physics analysis of events. Establishing early contact with the online group will go far in easing this transition into realtime application of the code.

5. *Other Algorithm Groups.*

    (a) *Calibration and Alignment.* An intimate connection with this group is absolutely essential in attaining the efficiency and precision necessary to extract the highest caliber physics from our data samples. Methods to populate and access the calibration databases will have to be developed.

    (b) *Silicon.* Methods to access the silicon detector hits and tracks (if found independently) must be available. As the responsibility for determining vertices now falls under the aegis of that group, methods to access the vertices found in an event need to be in place. The ability to refit tracks with a vertex constraint should exist, and protocols for handling this iterative procedure need to be developed.

    (c) *Muon.* Methods to access and extrapolate tracks found in the muon system need to be available, in order to associate and merge them with central detector tracks (and perhaps calorimeter information as well). (This holds whether this is the responsibility of the global tracking group or the later global reconstruction and physics objects groups.)

   (d) *Calorimeter.* Association of tracks and calorimeter objects will probably fall under the purview of the global reconstruction and physics objects.

Summary

The scintillating fiber track reconstruction software project is well underway, with a first version of the TRF++ code in place. Short term goals in addition to optimization of the current design and implementation include:

1. *Implementation of fiber clustering.*
2. *Connection to more realistic MC.*
3. *Extension to include silicon hits.*
4. *Additional class development.*
   (a) *Nonuniform field propagators.*
   (b) *Additional checker and filter classes.*
   (c) *Improved finders.*
5. *Development of track–finding paths.*

Completion of these tasks should result in code which is "adequate" for offline tracking reconstruction. Of course, much additional time and effort will be needed to properly debug and optimize the final program.

Available Manpower

The following individuals are currently working or have expressed interest in working on the track reconstruction software along with an estimate of their FTE commitment to the project: David Adams (100%), Norman Graf (75%), Jadzia Warchol, Drew Baden, Bill Cobau, Sarah Eno, Nick Hadley, Shuichi Kunori, Helio da Motta. Efforts are currently underway to recruit additional manpower.

List of D0Notes relating to the Scintillating Fiber Tracking

1. 1112 Performance of the D0 Sci Fi Tracking System.
2. 1177 Pattern Recognition in the D0 Upgrade Tracking System.
3. 1180 Pattern Recognition in the D0 Upgrade Tracking System 2: . Local 3D Supercluster identification
4. 1437 Efficiency and Resolution of a Double-layer of Scintillating Fibers.
5. 1619 Optimization of the D0 Step-1 Fiber Tracking System.
6. 2269 VLPC Characterization for the DZero Upgrade (DPF '94).
7. 2280 Performance of a Large Scale Scintillating Fiber Tracker . using VLPC Readout (DPF '94)
8. 2479 Cosmic ray test results of the D0 prototype scintillating fiber tracker.

9. 2665 Efficiency and Resolution of a Scintillating Fiber Doublet.
10. 2779 Performance of a Scintillating Fiber XUV Superlayer.
11. 2799 Performance of a Scintillating Fiber XUV Superlayer II: . Clustering without Gaps
12. 2893 A First Use of a Fast Tracking Algorithm in the D0 Upgrade Detector.
13. 2895 Timing XUV and XUVX Superlayers.
14. 2918 Adding pads to XUV_X and XU_XV Scintillating Fiber Superlayers.
15. 2958 Finding Tracks.
16. 3001 Light output in Scintillating Fiber Simulation.
17. 3056 Scintillating Pad Detectors.

4.1.4 Silicon Tracker Reconstruction

Overview

The Silicon Tracker in the upgraded DØ detector provides critical functionality for the experiment in Run II. Since it is not merely an improved version of a Run I detector, the whole body of software needed to analyze its data has to be written from scratch. The Silicon Algorithms Group is responsible for this task.

The Silicon Tracker produces digitized measurements of the charge deposited on its microstrips from the passage of charged particles through the silicon wafers. This information is used together with knowledge of the detector geometry to reconstruct each charged particle's track and energy loss as it passed through the detector. The tracks and dE/dx losses are needed for particle and parton identification and the track curvatures from the 2 T magnetic field are used to measure the particles' momenta.

It is a complex task to process the charge information from the Silicon Tracker into particle identities and momenta. The following list divides this task into natural subprocesses, and so provides a global overview of how we are working towards achieving this goal:

1. *Clustering.* Reconstruct space point "hits" by clustering the digitized charge depositions from the silicon microstrips.
2. *Tracking.* Reconstruct track stubs within the Silicon Tracker barrels and disks.
3. *Primary Vertexing.* Find the z positions of the primary interaction vertices.
4. *Secondary Vertexing.* Reconstruct displaced vertices in jets for heavy quark identification.
5. *Alignment.* Use reconstructed tracks to measure the relative alignment of the individual silicon modules.
6. *Calibration.* Use information from the online monitoring systems to improve the resolution and efficiency of the above tasks.
7. *Detector Geometry.* Create and maintain a database of the detailed geometry of the Silicon Tracker.
8. *Monte Carlo Model.* Fully simulate the operation of the Silicon Tracker so that input from a parton level generator is converted into digitized charge depositions on microstrips.
9. *Level 3 Trigger.* Do items 1–4 on this list (using items 5–7) fast enough and if necessary in a less efficient manner, so that they can be used online.

We are or will be interacting closely with the following groups: Fiber Tracker Algorithms (this group is responsible for the global tracking); Particle ID (we provide tracking, vertexing and energy loss information); Alignment and Calibration; Monte Carlo; Level 3 Trigger. All our work takes place under the umbrella of the Run II Algorithms Group. We are also working in partnership with the following hardware groups: Silicon Tracker (who provide us with geometry, alignment, and calibration information); Level 2 Impact Parameter Trigger Group (we provide the Monte Carlo model for these studies); and Silicon Test Beam (who will provide us with charge deposition information).

We now discuss in more detail the goals for the silicon algorithms software, and include status reports on the work in progress.

Long Range Objectives and Current Status of Each Project

1. *Clustering*

We are developing a sophisticated clustering algorithm to convert the digitized charge measurements from the microstrips into three-dimensional hit positions. The SVX II chips read out the digitized charge from all microstrips above an applied threshold, and they also provide the charge from the neighboring microstrips. This extra information is needed to improve the centroid position accuracy.

The charge clustering package will be run offline on the raw data and on the Monte Carlo simulated digitized output. A first pass of the hit clustering will be done before pattern recognition can start. However, the final hit positions will only be determined during the track fitting process, because microstrips with charge deposited on them from different particles will often be contiguous due to the high density of tracks within jets near the beamline. Since cluster splitting and merging will be iterated on whilst fitting tracks, it will not be possible to perform hit clustering online and then only record to tape the reconstructed hit positions.

Several steps are involved in optimizing the clustering algorithm to obtain high efficiency pattern recognition and good resolution for the reconstructed vertices. The charge distribution between neighboring strips is a function of the track incident angle in both theta and phi, the silicon strip orientation with respect to the magnetic field direction, the strip pitch, the overlap between neighboring silicon wafers, the presence of any dead or noisy strips, and aging from radiation damage. Some of these effects will be measured using test beam data and the results used to tune the algorithm before the start of Run II. Others can be simulated and corrected in the algorithm, for instance by turning off a representative sample of strips in the Monte Carlo model before performing clustering. A database of dead and noisy strips will be kept as a function of run number, so that detrimental effects from these causes can be overcome.

We have a simple clustering algorithm encoded in `FORTRAN` as part of the existing UPG_GEANT Monte Carlo program. This algorithm produces a hit position from a pulse-height-weighted centroid of a set of contiguously hit strips. The shift in the reconstructed hit position from the drift Lorentz angle has yet to be incorporated and no account is yet taken of incident track angle, or of dead or noisy strips. We are actively researching the details of what other experiments have done to optimize their clustering algorithms. We will soon be able to use results from the Silicon Tracker test beam (Spring 1997) to tune our Monte Carlo model of the distribution of charge on microstrips, and this improved model will then be available for optimizing the clustering algorithm.

We have started to design the hit classes needed for the clustering algorithm when we implement it in `C++`. A preliminary version can be found from the Silicon Software web page.

2. *Tracking*

The Silicon Tracker needs a stand-alone tracking package so that we can test the operation of the detector after construction and installation, before the start of Run II, and so that first pass alignments obtained at the start of the run can be fed back into the Level 3 trigger to increase its efficiency. We will test the tracker operation using cosmic rays as soon as it is assembled, and again after installation inside the Fiber Tracker to find out if any damage was caused during insertion. Tracking with cosmic rays will also be done to start to align the supermodules (barrels and disks) and component parts (ladders and wedges) once the whole central tracking system and magnet are inside the DØ calorimeter. The stand-alone tracking package could form part of the global tracking package for DØ, if it is found that global tracks are best constructed by linking track segments from each detector. However, other approaches to global tracking may prove to be more efficient; this will be determined by the Fiber Tracker Algorithms Group, which has responsibility for global tracking.

The main task in designing an effective stand-alone silicon tracking package is to achieve efficient pattern recognition. Since there are only four layers of silicon in the barrels, with gaps between the barrels for the disks, and small dead regions on the barrel ladders and disk wedges themselves, there will not be any superfluous hits to allow for inefficiencies. Some tracks will start with only three hits on them. We are working on a temporary pattern recognition and tracking package, by interfacing an existing one written by members of DØ in the `C` language for the STAR experiment at RHIC to the output of UPG_GEANT, in order to finalize the tracker design. Issues still open include the number of barrels (6 or 7), whether there should be 4 double-sided layers in the central barrels instead of 2 (to be partially paid for by eliminating a barrel), and whether the extra double-sided layers should have 2° or 90° stereo angle. This package will also help us to understand the pattern recognition efficiency as a function of the hit multiplicity on a track and of the local track density.

3. *Primary Vertexing*

The Silicon Tracker has a resolution on the primary vertex position of 0.5–1.0 mm (with 2° stereo microstrips in the barrels) or 100 μm (using a 90° stereo angle). In Run II with a beam crossing time of 396 ns, we expect an average of $\sim$6 primary interactions per beam crossing. The crossing region is extended along the beamline in a region with $\sigma \sim 25$ cm. The primary interaction z positions have to be found online in the Level 3 trigger, since that information is needed to generate for instance an electron trigger, together with energy measurements from the calorimeter and tracking information. Offline we also need the primary vertex positions for particle identification and physics analyses. We will develop an algorithm for identifying the positions of primary vertices which uses the stand-alone silicon tracking package. Later, we will refine these positions with the global tracking package. Global tracking needs to be used since it is important to associate all high p_T particles with the correct primary vertex.

4. *Secondary Vertexing*

Identifying secondary vertices in jets is the flagship purpose of the Silicon Tracker. We intend to develop several independent secondary vertex finding algorithms, based initially on the experiences of other experiments. It will be important to have more than one algorithm in order to cross check the results. The algorithms will tag jets as being from b quarks, c quarks or light partons ($u/d/s/g$). They will also measure the displacement of the secondary vertex from the primary one, and this distance will be used to determine lifetimes of inclusive and exclusive heavy flavor processes. We will reconstruct secondary vertices online in the Level 3 trigger, and this will be especially valuable if there is no Level 2 impact parameter trigger. Online identification of b jets will allow selection of singly and doubly tagged alljet events for study of many b physics processes and also some hadronic top and new physics ones.

The secondary vertexing algorithms will be refined and optimized most carefully to be run as part of the offline reconstruction as well as online. This effort will require a long period of study both before the start of Run II and during the first year with collider data. We have started to review the algorithms used by CDF, OPAL, SLD and other experiments, and have begun design work on the `C++` classes needed to implement our own algorithms.

5. *Alignment*

Alignment coefficients will be incorporated into a database of the detector geometry and updated periodically. The starting point will be information from the optical surveys undertaken during construction of the tracker, followed by information from cosmic ray tracking, then finally tracking from $p\bar{p}$ collisions. The data will include three translations and three rotations for each rigid body in the tracker (ladder, wedge) and from these we will derive global translations and rotations of the larger modules (barrel, disk). Finally the position of the whole Silicon Tracker with respect to the Fiber Tracker will be measured and included in the geometry database. This task is a large, critical job which must be done very rapidly once Run II starts in order to make the Level 3 tracking trigger efficient and to do first pass physics analyses. Continuity for updating the alignment will be essential to keep it at maximum quality throughout the run, since the tracker may slowly relax and sag inside DØ, and also if the outer detectors are worked on during the run then the vibrations may cause the inner detectors to shift slightly. To achieve this continuity, a dedicated university or laboratory group will need to have responsibility for the alignment of the Silicon Tracker. Studies of the alignment precision required for construction have of course taken place already, but they need to be updated soon to understand what is needed and achievable offline.

6. *Calibration*

We will need to include information from the online slow monitors, such as readout chip and silicon detector voltage levels, accumulated radiation dose, dead or noisy channels, temperatures, shutdown periods, and so on, into the Monte Carlo model of the Silicon Tracker and update the information throughout the run. Some of the information will also be needed when the raw data is unpacked and the charge measurements from the microstrips are clustered into hit positions. We therefore need to understand well before the start of Run II which pieces of information are the most important for high resolution tracking and vertexing, and thus how precisely they need to be measured and how often they need to be updated in the databases.

7. *Detector Geometry*

For Run II there will be one detector geometry database which will be used in both the track reconstruction and the Monte Carlo model. The geometry will be described using GEANT4 structures. The information will include not just the active regions of the detector, for generating hit positions from clustered charge from microstrips, but also all the inert material of the tracker, for simulating multiple scattering, brehmsstrahlung, pair production, and so on. During the building of the Silicon Tracker, all inert material must be fully measured and documented, since once the tracker is closed up and installed it will be very difficult a posteriori to work out what was actually built. We will use photographs of the tracker as it is constructed to get the modeling of the cabling correct, for instance, as it will be highly nonuniform in phi.

We have already created a preliminary database of the tracker geometry (except the H-disks), implemented in FORTRAN and GEANT3 for UPG_GEANT. This is being used to understand and optimize the tracker design and as input to work on the pattern recognition and track fitting. Figure 5(a) shows how the active regions of the barrels and disks appear for a 6 barrel module design, and Fig. 5(b) shows the first inert material included in the model, a barrel bulkhead. Much of this database will be transfered directly into the full Run II Monte Carlo package D0GSTAR.

8. *Monte Carlo Model*

The Monte Carlo model of the Silicon Tracker takes as input the charges and track trajectories of particles produced from a generator such as PYTHIA or HERWIG and projects them through the active and passive material of the tracker, as contained in the detector geometry database. When a particle passes through an active region, such as the silicon of a ladder or wedge, then

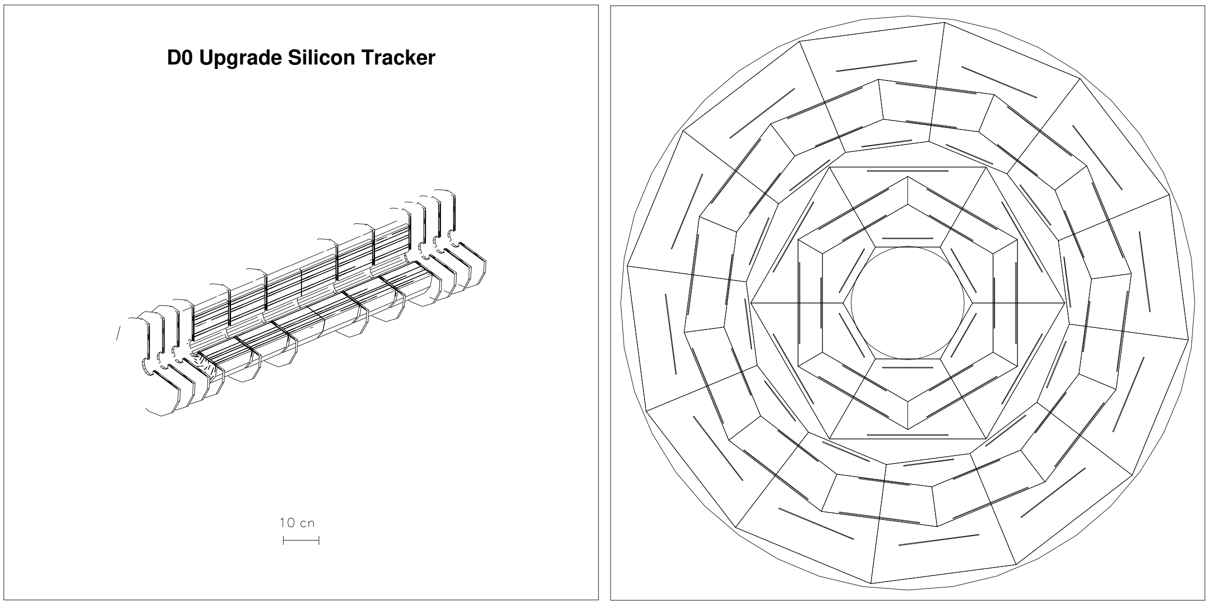


Figure 5: (a) View of the GEANT model of the Silicon Tracker. (b) Preliminary implementation of the Silicon Tracker barrel bulkhead in GEANT.

it simulates the generation of electron-hole pairs, and calculates the drifts of the electrons and holes in the silicon caused by the applied electric and magnetic fields. From this information it reproduces the analog signals seen on each microstrip, and then digitizes the information using the same digitizing algorithm as used in the SVX II readout chip. The Monte Carlo model generates the appropriate multiple scattering, brehmsstrahlung and pair production as these particles pass through the active and passive media of the tracker. Many of these processes will be handled by the GEANT package, but the specific details of the generation of the electron-hole pairs in the silicon, drifting in the fields, generation of the analog signals and digitization of those signals are the responsibilities of the Silicon Algorithms Group.

Several years ago we obtained from ALEPH a detailed Monte Carlo model of silicon microstrip detectors, and this is implemented in the UPG_GEANT package for DØ's Silicon Tracker. We are updating it to work for 90° stereo microstrips ganged together in non-neighboring groups, and are also working to include a realistic model of Landau fluctuations. We are now fixing many bugs and documenting it, so that it will be useful longterm.

9. *Level 3 Trigger*

The Level 3 trigger will contain as much as possible of the full offline reconstruction package, including tracking, vertexing, alignment and calibration. We have not yet started to design the trigger as we need to first implement a working version of the algorithm packages themselves before knowing how fast they are and how much of them can be used online.

Summary

The Silicon Algorithms Group is actively pursuing the many tasks outlined above in order to have a mature tested reconstruction package ready for the next run. We are recruiting many people, holding weekly meetings, and documenting our progress on the DØ web pages. Our short-term milestones include incorporating the first inert material in the detector geometry database using GEANT 3.21 (which is new to DØ) by mid-January, followed by a series of small changes needed to make the database geometry match the current tracker design. By the middle of January we also expect to have a first working version of the temporary tracking package in order to study urgent pattern recognition design issues. We have begun work on a Kalman vertex finder and expect to have it running for preliminary evaluation by the end of February. For the middle of March, we will have the hit classes and geometry classes defined and implemented, and will also have an initial version of the silicon-specific tracking classes designed. The first full versions of the clustering, tracking, primary and secondary vertexing algorithms, and the Monte Carlo model of the tracker with its full geometry in GEANT4, have to be implemented in `C++` by the end of 1997.

4.1.5 Calorimeter Reconstruction

Signals from charged and neutral particles that pass through or shower in the calorimeter are recorded for each event. The reconstruction software is used to convert these signals to energy and transverse energy (E_T) and to then find electrons, photons, jets, muons and missing-E_T in the event. The software will be used for online and offline reconstruction of the physics observables, event display graphics and Monte Carlo simulations.

The calorimeter itself is not being changed from the Run I design. Therefore the geometry database, which contains the positions of the readout cells in the calorimeter will not change, except for the conversion to a C^{++} compatible database. However the readout electronics will change because of the increased number of bunches in the accelerator, therefore giving less time between crossings. The method for correcting the readout cells for their pedestals and relative gains will have to be redesigned according to the new hardware specifications. Also in Run II, there will be preshower detectors, situated just inside the calorimeters, and an inner magnetic field. Both of these will affect the distributions of energy from particles traversing the calorimeter. Therefore the calibration method from signals in the readout electronics to energy will also been redesigned.

As a starting point, the jet reconstruction software used in Run I is being converted to C^{++} in a prototype program and is discussed in detail below. Much of the design work of the framework and structure of the software in this prototype project will carry-over to the final design of the calorimeter reconstruction code. The lessons learned from this prototype project will also be applied to the design and structure of many other detectors and software structures in the experiment. A first iteration of this software is expected to be finished by the end of February 1997. Then extensive testing of the software and comparison of results will take place.

The short-term plans for the calorimeter reconstruction software includes work on the jet reconstruction prototype. This will include work on the geometry and jet algorithm software along with the overall event interface.

In the future, the calibration software will be designed, to allow calibrations and corrections to the calorimeter signals to be made at the appropriate places. The actual geometry database will need to be written in a format so that the graphics, Monte Carlo, and reconstruction software can all use the same database. The preshower and inter-cryostat detector software will be designed. For all parts of the calorimeter reconstruction software, a list of requirements will be made and then there will be extensive work done on the class design before the final coding.

Run I Jet Reconstruction Prototype

This project, which is near completion, entails the analysis and design of the structure of the software used to reconstruct jets. The class design and inter-class relationships have been modelled extensively using the Rational Rose design tool in the OMT notation. During the design process, scenario and event trace diagrams are also made for each part of the software. The input information is an event vertex, the physical geometry of the calorimeter cells and the energy in each calorimeter cell from Run I data. The main parts of the software that are currently being coded are the calorimeter hardware classes and the clustering algorithm classes.

In the calorimeter hardware class design there are *Cell* and *Tower* (cells in the same η and ϕ position which are grouped longitudinally) classes which contain information about the position, energy and adjacent neighbors of the cell or tower. Physical positions of the cells and towers are obtained from the *CalorimeterGeometryDatabase* class. The actual database file is read only when the information is actually used, saving many time- and memory-consuming instantiations and allowing different client classes to obtain only what they actually need from the database. A first iteration of the prototype program containing these classes has been written and is currently being extended to be made compatible with other parts of the reconstruction software.

The cluster algorithm classes have the responsibility for creating clusters of energy depositions. The design is being made general so that it can be easily modified to add or change the jet algorithms and so that much of the design can be kept for the final Run II reconstruction. The class which does the actual work of determining which towers (or cells) belong to which cluster is *ClusterAlgorithmBase*. There are several subclasses of this class which define different types of cluster algorithms. The *SeedUsingClusterAlgorithm* is currently being coded. It uses a seed tower (a tower with an energy above a minimum threshold) as input and then does the actual clustering around the seed tower. The *MergeSplit* class will then perform the merging and splitting algorithm on overlapping energy clusters. A first iteration of the prototype program has been written, which reads in cell energies from a simplified datafile and forms energy clusters.

The other main part of the jet reconstruction prototype software are the classes which form the jets. After all clusters of energy are found, the *JetCalculator* class takes the energy clusters into jets. The energy and direction of the jet is defined at this point. This design of this part of the software is being finalized and will be coded in the near future.

4.1.6 Muon System Reconstruction

Overview

The offline muon software needs to accomplish three basic tasks:

- Raw Data Unpacking. The input file contains blocks of raw data corresponding to VME crates. The data need to be sorted according to detector modules and their associated channels. In this form, the data can be used as input for hit processing, trigger simulation, and hardware monitoring.
- Hit Processing. At this stage the raw data is converted to distance and time measurements, using calibration constants from the database. Processed data is used in track reconstruction, detector calibration, and alignment.
- Track Reconstruction. Pattern recognition is performed to identify track segments, measure momenta, and match tracks to scintillator times.

Current Status of Run II Code

We have completed a preliminary design of the software for raw data unpacking and hit processing. We have identified the classes needed and their relationships, and we have a model for the flow of data. However, no C++ code exists at this time. We have not done any design of the track reconstruction software, since we expect that this will be part of larger software package used by all the detector groups.

Future plans

We are ready to start translating our models into code. We intend to follow the lead of the Calorimeter group, and modify their C++ framework program to suit our needs. Since no DSPACK interface between Geant and C++ exists at this time, to start with we will read in dummy raw data from an ASCII file. This should be adequate for the initial stages of code development. We will begin with raw data unpacking. This is the logical place to start, since the D0 Geant program produces simulated raw data for the muon system.

Available Manpower

The labor available for writing Run II muon code includes the following physicists and their FTE fractions: Rich Markeloff, 50%; Paul Quintas, 25%; and Vladimir Sirotenko, 25%. In addition, we have a new colleague from Russia, Alexander Kostritsky, who will be joining our group. Dave Hedin and Mike Fortner will be providing input based on their experience with the Run I software, but will not actually be writing Run II code.

4.1.7 Level 3 Algorithms

We have postponed consideration of Level 3 algorithms as such, because the plan is to use the offline algorithms wherever possible, developing speedier versions when necessary. But the leaders of the group – Amber Boehnlein and Dan Claes – are working on simulation, verification, and framework issues.

4.1.8 Calibration and Alignment

The Run II Calibration and Alignment subgroup is responsible for aligning and calibrating each individual detector. These detectors include the silicon tracker (barrels and disks), the fiber tracker, the calorimeters (central and endcap), the central preshower detector, the forward preshower detector, the muon detector (wide angle, forward, $A\Phi$ counters), the Level 0 counters and the solenoid. The group is responsible for defining the necessary alignment/calibration constants, for developing the algorithms and writing the code to determining those constants, and for specifying the triggers, filters and data sets to be used. It is also responsible for developing the interface to the database(s) used to store the constants. Finally, the group has responsibility for global, interdetector alignment and calibration. These responsibilities include constants used both offline (in RECO) and in the Level 3 trigger. It is planned that the membership of the group will include a contact person from each detector (silicon, fiber tracker, calorimeter, central preshower, forward preshower, and muon) as well as from the Level 0 and solenoid groups. The overall coordinators are John Hobbs and Harry Melanson.

The first step in alignment/calibration is to define the geometry of each detector, which specifies the idealized (ie. designed) positions and orientations of the detector components. To define this geometry, a hierarchical approach will be used, where first a global description of the detector is defined, followed by more and more detailed refinements, until each individual channel is specified. As an example of hierarchical geometry, one can define the silicon barrels as a collection of concentric cylinders of given radius. Planes would then be defined relative to these cylinders, to represent the silicon ladders. Next, wafers would be described relative to these planes, and finally strips would be specified relative to the wafers.

The alignment of a detector maps the specified geometry to the actual, physical positions and orientations of the real detector. We have identified two generic types of alignment: *geometrical*, which represents linear, decoupled transformations, and *distortional*, which allows for non-linear, coupled transformations. With the hierachical nature of the geometry, it is possible to perform distortional-type alignments using a series of geometrical-type corrections. The choice of which type of correction will be used will depend on the details of the particular detector.

The calibration of a detector deals with all non-geometry related transformations used to convert detector data into useful information. Examples include pedestals, gains, drift velocities and t_0's.

The Monte Carlo also needs to use geometry and calibration constants to model the detector. We anticipate the need for an intermediate program (ie. PREGEO), which will take the geometry defined above and convert it into the appropriate Monte Carlo format (ie. GEANT volumes). This program should be able to use as input either the idealized geometry or the aligned geometry. This will ensure that the Monte Carlo will be able to simulate the actual, physical detector. It also allows the Monte Carlo to be used to develop and test the alignment/calibration algorithms and code.

Table 1 lists the key milestones for the Calibration and Alignment subgroup.

Table 1: Milestones for Calibration and Alignment Subgroup

Date	**Description**
April, 97	Define hierarchical geometry for each detector
June, 97	Define DB interface
Sept, 97	Complete geometry class design Implement prototype DB interface to flat ASCII files
Dec, 97	Generate "misaligned" Monte Carlo data Complete design of overall Cal/Align program
Mar, 98	Complete design of detector Cal/Align algorithms Define triggers, filters, data sets required Implement Version 1 of Cal/Align Program
June, 98	Implement Version 1 of detector Cal/Align algorithms Implement interface to actual DB Complete design of Energy Scale Calibration
Sept, 98	Implement Version 1 of Energy Scale Calibration

4.2 Manpower, Schedule and Software Review Procedure

In order to estimate the manpower needed to deliver a first version of the reconstruction program by the fall of 1998 we have itemized the required tasks for each subgroup and specified by what date each task must be completed. A summary of the schedule in three months intervals is shown in Tables 2 and 3. Major milestones are expected in June 1997 when reconstruction code for subsystems should be completed, and in December 1997 when global tracking and vertexing will become available and the first design of packages to identify physics objects (ie electrons, muons, etc.) will be under review. The schedule is not rigid and will be revised as time goes on but we cannot allow the final goal to slip: all major components must be ready by the fall of 1998 when testing of the full reconstruction program will begin.

Each subgroup has estimated the amount of manpower it needs in full time equivalents (FTE) for designing, writing and testing code over the coming year: 9 for calorimeter, 7 for Scintillating Fibers, 7 for Silicon Vertex, 6 for muon subsystem and 5 for calibration and alignment. Global tracking is expected to be developed as a joint effort between the Scintillating Fibers and Silicon Vertex groups. The total estimated number of FTE's is 34. This number may have to increase further by next fall when additional subgroups will need to be formed to design and write code to identify physics objects.

To meet the planned schedule it is essential to closely monitor progress and have detailed design and code reviews. Six code review teams of 3-4 experienced programmers have been assigned the task of monitoring progress of various components. The review teams will produce written reports on whether the design and code meet standard requirements and details of tests done to demonstrate the code performs as expected. Those reports must be completed by the dates indicated in the schedule. Before any code is inserted into the software library for distribution it will have to be approved by the review teams and they may request modifications and additional tests before approval. A given review team excludes people working on the code under review but includes people working on other aspects of the reconstruction program. We have also included in the review teams experienced programmmers involved in D0 software activities unrelated to the offline reconstruction (such as online software, software tools, etc.).

5 Monte Carlo Tools

For Run II we shall continue to use the GEANT simulation framework because it is the world standard, debugged and extensively verified against data over a huge range of energies. The existing `UPG_GEANT` simulation incorporates all Run II detector elements and generates detector data in `ZEBRA` format. It has been and continues to be extensively used for trigger studies, optimization of detectors such as the forward muon system, development of the shielding, etc. Improvements to this simulation are underway (as described earlier) to incorporate the latest revisions in detector design.

Over the next six to nine months we plan to move to a new framework for geometry definition based on the CERN `TZ` package. This will enhance portability between platforms, ease the future transition to the C++ version of `GEANT` and simplify the task of maintaining the code. The new simulation will be called `DOGSTAR`. A pre-release version including the full geometry exists; work on the digitization phase is proceeding. The geometry input to `DOGSTAR` will also be used to define the input to the geometry classes used for reconstruction. Output data from `DOGSTAR` is currently in the form of `ZEBRA` banks but will switch to `DSPACK` once the Run II data format is stable.

We are also investigating the development of a fast DØ simulation based on the `MCFAST` package from the Fermilab computing division. This tool has already been used in the optimization of the tracking system. Additionally, our experience with Run I analysis has shown that fast simulations can be very useful; in the absence of an official fast simulation for Run I an undesirable profusion of unofficial versions has developed. Clearly this should be avoided in future. How best to integrate the fast and full simulations still needs to be studied.

Table 2: Schedule for Run II reconstruction program (part I): des=design, 1st=code 1st version, 2nd=code 2nd version.

	3/97	6/97	9/97	12/97	3/98	6/98	9/98
General							
D0 C++ framework	2nd						
Dspack I/O	1st						
central archive	1st						
OO Run I data		des	1st				
Graphics	1st		2nd				
Global RECO							1st
Calorimeter							
Calorimeter cells in Dspack	1st						
geometry classes	1st	2nd					
cells, towers	2nd						
clusters, jets	1st	2nd					
raw and raw to hit			des	1st		2nd	
Muon							
geometry objects	1st	2nd					
hits objects	1st						
local tracks	des	1st	2nd				
raw and raw to hit			des	1st	des	2nd	
Si Vtx							
geometry objects	1st	2nd					
hits objects	1st	2nd					
new Geant geom	des	1st					
local tracks	des	1st	2nd				
raw and raw to hit			des	1st	des	2nd	
Fiber Tracker							
geometry objects	1st	2nd					
hits objects	2nd						
local tracks	des	1st	2nd				
raw and raw to hit			des	1st	des	2nd	

Table 3: Schedule for Run II reconstruction program (part II): des=design, 1st=code 1st version, 2nd=code 2nd version.

Physics Objects	3/97	6/97	9/97	12/97	3/98	6/98	9/98
tracks, prim. vtx	des	1st	des			2nd	
2nd vertices		des	1st	des		2nd	
e,μ,$\pi^{\pm}$,...				des	1st		
K_s, Λ, B and C					des	1st	
b-jets					des	1st	
W, Z, psi						des	
Calib & Align							
DB interface		des	1st				
generate geant data				1st			
alignment program				des	1st		
det. calib. schemes					des	1st	
E-calib. classes						des	1st
Monte Carlo							
DSPACK interface	1st						
OO MC objects	des	1st					
create raw data		des	1st		2nd		
alignment tests				1st			

6 The Reconstruction Farms

It is expected that DØ will do near real time reconstruction of the event stream on the FNAL production farms. The events will be reconstructed on standard configuration farm nodes using CPS control software (or its replacement as provided by the Computing Division).

6.1 I/O Requirements

The I/O requirements of the reconstruction farm can be estimated from the expected event size and average event rate. With an event size of 250Kb, an event rate of 20Hz, and assuming the full raw data stream is written out with the reconstructed data, we estimate that the aggregate bandwidth of the reconstruction system must be in the 10-12Mb/s range. Since reconstruction time on any CPU likely to be available in the time frame of Run II will significantly exceed one second we can assume the I/O requirements to any single farm node are small enough to be handled by a standard ethernet connection, i.e. less than 500Kb/s. This, of course, assumes that only one process is running on each farm node.

6.2 Memory Requirements

In the absence of a completed reconstruction program the memory requirements of the farm nodes are difficult to estimate. DØ's Run I reconstruction program was approximately 40Mb in size. This, however, may not be a good indicator of code size for Run II. Increased complexity of the DØ detector, increased luminosity, and the shift from Fortran to C++ code base may all significantly effect code size.

In principle the shift to C++ could reduce overall memory requirements through the use of dynamic memory allocation. However, the use of dynamic memory could also increase memory requirements as the luminosity (and hence the number of interactions per crossing) increase. The reconstruction program will consist of several packages, only one of which will be active at any time. One could expect that the static memory requirements of an inactive package are small compared to its active memory needs. If that is true, the memory requirements of the reconstruction program will be dominated by the needs of the largest active package. The only part of the reconstruction program which currently exists in a C++ incarnation is the TRF++ package for the fiber tracking system. This code currently uses about 60Mb of memory when run on idealized data representing 5 interactions/crossing. Since this is not mature code and it has been run only on idealized Monte Carlo data, its requirements must be viewed as a lower limit only. The reconstruction memory requirements of other detector subsystems are not currently known, but it is safe to assume that the central silicon tracker will have comparable or larger memory needs.

From the above considerations the best estimate we can make at this time is that the farm production systems will need, at a minimum, 64Mb of user memory per active reconstruction process. The machines should be upgradable to at least 256Mb of memory per active process if necessary.

6.3 CPU Requirements

All estimates of CPU power in this section are made with reference to the TINY benchmark program used by the Computing Division for rating CPU power. All CPU ratings were either made by directly running the TINY program on the CPU in question, or taken from the comprehensive list of CPU ratings made at Fermilab as compiled by Steve Wolbers of the Computing Division.

As reference, the TINY ratings of several commonly used CPUs are listed in the table below.

CPU	TINY Rating
VAX 4000-60	11
VAX 4000-90	27
150MHz R4400SC	75
Alpha 200 4/233	100
Alpha 250 4/266	125
IBM 43p 133MHz	125
195MHz R10000	195

One way to estimate the CPU requirements for Run II reconstruction is to simply extrapolate from our Run I experience. DØ's Run I reconstruction program required about 40 seconds to reconstruct a single event on nodes rated at 21 TINYs (33Mhz R3000 Indigo), or about 830 TINY-secs/event. Folding in the expected 20Hz data rate then would lead us to believe we would need at least 16,600 TINYs of compute power to do the near real-time reconstruction of the full event stream.

During Run I DØ used about 25% of its farm capacity for reconstruction of Monte Carlo events or re-reconstruction of small event streams. The overall efficiency of the farm system varied in the 50-90% percent range during various portions of the run. The 90% number was only achieved near the end of the run and at the cost of very detailed tuning of the system. It is fair to assume that we will only achieve about a 75% duty cycle integrated over most of the run. Adding these two factors to the above numbers results in a final estimate of roughly 30,000 TINYs of compute power for DØ's reconstruction farm needs.

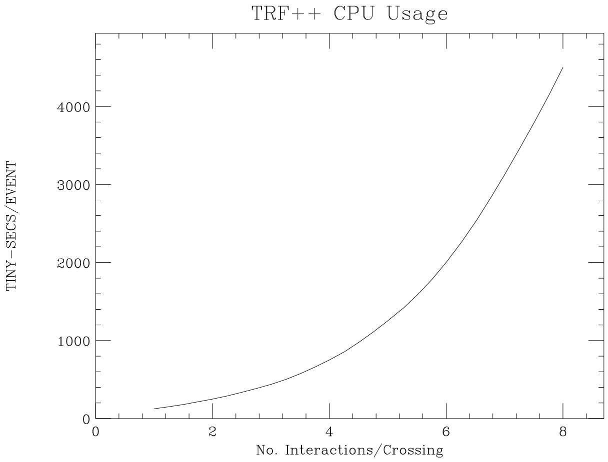


Figure 6:

The above is likely to seriously underestimate the CPU needs of the reconstruction system for several reasons:

- The number of 40secs/event was achieved for single interaction events. We do know that the execution time rose dramatically with increased luminosity (and hence the number of interactions/crossing).
- It is unlikely that the addition of a magnetic field will leave the reconstruction time unchanged.
- The code base is being changed from Fortran to C++. The actual impact of this change on execution times is currently unknown. However, some aspects of object oriented programming ,such as data encapsulation, which require method invocations for all data accesses will likely result in increased execution times.

Another, and probably more reliable estimate of the CPU requirements, can be made from the TRF++ program mentioned above under Memory Requirements. It is important to keep in mind that this is preliminary code with no provisions for noise, inefficiencies, multiple scattering, nonuniform fields, or any other real world complications. As such it is certain to underestimate final reconstruction times.

Extensive timing measurements have been made on the TRF++ code by David Adams. These timings were done on a 133MHz IBM 43p workstation. The code was compiled with IBM's native CSET compiler but without architecture specific optimizations (equivalent to the 125TINY rating in the table above). Figure 6 shows the CPU resources necessary to do fiber reconstruction as a function of the number of interactions/crossing.

The fiber tracking system is only part of the DØ reconstruction program, but it is currently the only portion in a functioning state. We will make the following assumptions to estimate the total reconstruction time for Run II:

- The central Silicon Tracker will require about the same CPU resources as the Fiber Tracker for reconstruction.

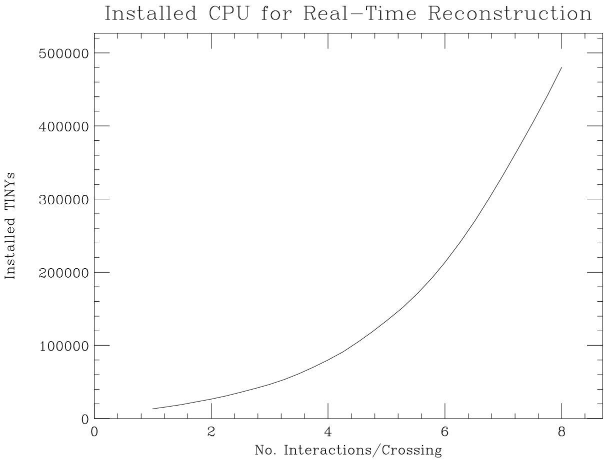


Figure 7:

- All other detector subsystems put together will require another increment of time approximately equal to the Fiber Tracker time.

Hence the available farm resources necessary to do full reconstruction will be about three times that shown in Figure 6. If we now fold in the same factors used in extrapolating the Run I reconstruction times (i.e. 20Hz event rate, 75% farm efficiency, and 75% farm utilization) we can find the total installed CPU necessary to do near real-time reconstruction of the full event stream. This result is shown in Figure 7.

Hence we can expect to need somewhere between 13,000-480,000TINYs of installed CPU to do near real-time reconstruction on the full event stream. If we take as a reasonable estimate something between 3-4 interactions/crossing, we get about 60,000TINYs. It must be be kept in mind that this represents a lower limit on the needed CPU.

It is necessary to determine if the CPU ratings as measured by TINY (a small, Fortran program) are actually representative of what we can expect with code like TRF++ (a large, C++ program). David Adams has now run TRF++ on several different CPUs at FNAL. The results of these runs are shown in the table below.

CPU	Compiler	Time	T/T(43p)	TINY(43p)/TINY
IBM 43p	CSET	993	1.00	1.00
IBM 43p	GCC	1496	1.51	1.00
IBM 370	CSET	2209	2.22	2.55
R4000 100MHz	GCC	3329	3.35	3.05
R4000 200MHz	GCC	1591	1.60	1.25
R10000 195MHz	GCC	701	0.71	0.64

The measurements were made for 100 events with 5 interactions/event. It appears that the TINY ratings track reasonably closely to the TRF++ timings. The difference in times on the 43p processor with the CSET and GCC compilers indicates that all the above estimates of CPU needs have uncertainties on the order of 50% depending on what compiler is used. If the architecture specific optimizations had been used, the TINY rating for the 43p would be 155 and the differences between the CSET and GCC results even greater.

Let us compare the estimate of 60,000 TINYs to the maximum CPU we can expect the lab to have installed by the beginning of Run II. The Computing Division purchased new farm systems in '96/97. The acquisition comprised about 15,000TINYs of CPU for near $1M. This included all associated hardware and I/O facilities. If we assume the laboratory will spend an additional $1M in each of '98 and '99 and that the average cost of the CPU over these two purchases will drop by a factor of two, then we could expect the lab to put an additional 60,000 TINYs online by the beginning of Run II. This farm capacity will have to be shared between DØ and CDF, who will have comparable needs, and possibly with ongoing fixed target reconstruction as well. It is therefore hard to imagine that DØ can have more than 35,000 TINYs available at the start of Run II. Our conclusions from these calculations are:

1. The farm procurement budget and the reconstruction program performance together will be the throttle on physics production from Run II data.

2. The current "best" estimates indicate a shortfall of approximately a factor of two in real-time capacity at the beginnning of Run II. It is possible to imagine that this shortfall could be remedied during the following fiscal year (with an infusion of $2.5M per collider experiment) but this amount has yet to be formally incorporated in budget planning by the CD or the Lab. Also, we have to note that the estimate of 60,000 TINYs is (1) based on a rather optimistic estimate of reconstruction performance and (2) scales with the 20 Hz number for Level 3 output – a number which is not a major physical limitation in the online system.

3. The action items for DØ with regard to this problem must be to have this budget shortfall and its consequences clearly understood by Lab management, and to place a high priority on reconstruction program performance throughout its development.

6.4 Remote Computing

Several DØ institutions besides Fermilab have provided large computing farms for Monte Carlo simulations for Run I analyses and upgrade studies. In particular, the Brazilian group has both a very large farm and a well-developed client server software suite which are currently used for Monte Carlo production. We envision for Run II that essentially all CPU-intensive Monte Carlo projects will run on remote farms modelled after this one.

6.5 Software Requirements

Experience from Run I has shown that efficient operation of our production systems requires that some minimal debugging capabilities be present on the farm nodes. Full graphical debugging tools are not necessary, but we must have the minimum capability provided by debuggers like dbx.

7 The Analysis Environment

7.1 Computing Model

The major change in DØ's computing model for Run II will be the shift from an environment where computing is done on distributed platforms to one where most computing is done on central machines with local data access. Only limited end analysis (i.e. histogram generation from limited ntuples) is expected to be done on distributed workstations.

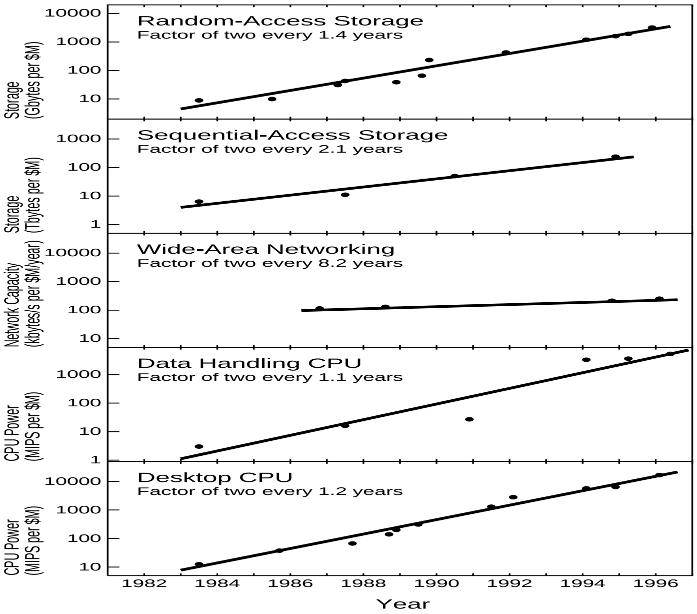


Figure 8:

The impetus for this shift is the increased amount of data that DØ expects to collect during Run II. The factor of 10 increase in data will require a comparable increase in resources associated with the analysis of that data. While we can probably expect to engineer a factor of 10 increase in CPU capacity, it is very unlikely that comparable increases in network bandwidth will materialize. Thus timely analysis of the data will require that most computing be done with local disk access to data. In addition, we believe that efficient management of the computing resources requires a centralized computing model.

We envision a system comprised of central SMP machine(s) of ~40,000TINYs of compute capacity. The system will require ~10TB of disk capacity and a throughput from disk to CPU of ~300MB/S. This system will need access to HSM storage of ~100TB and shelf storage of ~150TB/yr. These numbers will be examined in detail in the following sections.

Although data analysis requirements will increase by a factor of ten, that does not imply that resource requirements for interactive type activities will also increase at the same rate. Consequently we envision a segregation of batch and interactive activities. This will allow batch systems to be used near their maximum capacity without adversely impacting interactive use by users. This segregation may be done either through physically separate systems, or through resource allocation software. Similar segregation of resources may be desirable on the workgroup level.

It is expected that access to the central systems from the desktop will be via a mixture of X-terminals, workstations, and PCs. It is not intended that workstations will have direct access to the full central disk set as was true in Run I. I.e. we will not export data disks from the central systems to the desktop. Necessary access from the desktop will be done via ftp.

In the following sections several assumptions are made about the evolution of price/performance ratios for various aspects of computing. The numbers in question are taken from the data in Figure 8.

7.2 Storage Requirements

Assuming we continue to process data the same way we did in Run I we can expect to see the data accumulations shown in the table below. These numbers assume an event rate of 20Hz.

Data Type	MB/Event	TB/year
RAW	.250	158
STA	.300	189
DST	.050	32
μDST	.005	3
Total	.605	382

No duty cycle has been folded into these data accumulations. The expected 20Hz event rate is not an actual hardware limitation, but a target determined by offline resource limitations. Hence, it is expected that actual data rates will be adjusted to keep the offline pipelines filled. Effectively the duty cycle will be 1.0 as far as the offline systems are concerned.

7.2.1 Serial Storage

We can extrapolate from our Run I serial storage usage to estimate what total serial storage might be needed for Run II. In Run I production of STAs, streaming, and various data set replications resulted in a total amount of data stored on tape that was about a factor of 5 greater than the size of the RAW data alone. Such replication in Run II would result in about 800TB/yr of data on tape. If our average occupancy of tapes is near 70%, then we are in the range of 1 PB/yr of serial media storage.

This type of data replication needs to be judged in light of the potential costs of this much serial media. All the current generation of serial storage devices (NTP, DLT, Redwood, Mammoth) have media costs that are in the $3-5/GB range. DLT 7000 technology is currently the cheapest at about $2.80/GB. At current prices we would then be facing around $3M/yr in media costs. This is clearly not within the realm of possibilities.

As with other aspects of computing we can expect that there will be a decrease in the cost of serial media storage before the beginning of Run II. However, the change in media costs is more difficult to predict than the change in CPU or disk costs. The current generation of high density serial media actually represented a significant increase in media cost/GB compared to video grade or even data grade 8mm tape. In addition none of the new generation of storage devices have a mass market counterpart which could drive prices down. Some manufacturers are also maintaining exclusive distribution rights for their media. As a point of comparison, the price the laboratory paid for 8mm video grade tapes dropped by about a factor 3 over the period from '91 to '95. This drop was caused by demand from the consumer electronics industry, not by demand from data storage applications. Factors of 5-6 decrease in media costs within the next few years would be an unrealistic expectation. We should be prepared for the possibility that we will not even see a factor of two decrease by the first year of data taking.

These costs lead to several conclusions regarding our offline data processing:

- We must make the maximum use of media. Sixty to seventy per cent occupancy rates on serial media will be unacceptable.
- We will not be able to afford the replication of data produced by streaming of STAs.
- No significant copying of RAW or STA tapes will be done for use off-site.

- Even the production of STA tapes is in question. We should consider the possibility of writing only DST output from the reconstruction program.

Since the potential impact of tape costs on our offline processing plans is significant, it is important that we get an estimate as soon as possible of what operating funds will be available for media.

7.2.2 Disk Storage

Ideally we would like to store all data in a random access object database. Exactly one copy of each data object would be retained and all objects would be accessible to users on an equal basis. In practice we cannot keep all data on random access media simply because of cost. We will have to settle for a mixture of a frequently accessed disk resident objects and less accessible tape resident objects. If we maintain the object nature of the store this will still allow us the flexibility to change the contents of the disk resident set without reprocessing the entire data stream. The CAP system developed by the Computing Division was intended, in part, to provide such a functionality. Its functioning is dependent upon identifying a cardinal, disk resident, data set from which most analysis can be done or at least, from which event selections can be made.

The smallest acceptable data set we have identified for usage in Run I has been the μDST set. This set was not arrived at by combining the needs of all the analysis efforts, but rather as a grudging compromise between groups. It is inadequate for some active analysis efforts. As such, it can be viewed as the minimum size data set that will be acceptable to have resident on disk.

As listed in the above table, we can expect to accumulate about 3TB of μDSTs per year. In Run I we have typically used an amount of project/tmp disk space on the analysis clusters comparable to what was used for μDST storage on D0FS. It is reasonable to assume that this project space will scale the same way the data does. Hence we also expect to need about 3TB of project space per year. An additional amount of space was used in Run I on D0FS for various logistic and processing activities. This requirement may well not scale as quickly with data set size. We will assume that the logistic operations will require about 1TB of disk per year. This gives us a total requirement of 7TB/yr of disk space.

The Seagate Elite-23 23GB drives are expected to be available soon at a cost of around $3,000. Historically there has been a factor of two improvement in price/performance on random access media every 1.4 years. It should be reasonable to expect that disks will be in the $50/GB range in mid '99 when the first of these disks will need to come online. Assuming the disks are at least 50GB/drive capacity, we would need 140 drives/yr. This number of drives is easily handled by large SMP systems. The cost of such drives should be around $350K/yr.

7.2.3 HSM Storage

Ideally the entire DØ data set would be resident in robotic storage so that no human intervention was ever required for data access. If we assume that our entire data set consists of one and only one copy of the RAW, DST and μDST sets (i.e.no STAs), then from the above table we would need about 200TB/yr of robotic storage.

The cost of robotic storage currently varies widely depending exactly what serial media technology is used. Costs for 15TB of storage vary from around $175K at the low end to around $1M at the high end at this time. Assuming a factor of two decrease in cost between now and the beginning of Run II when the first of this storage must be in place, we would expect this to cost in the range of $1.2-6.5M/yr. This is a high cost especially since the bulk of the RAW data will only be accessed once.

From Run I we do know that frequent accesses to RAW data are necessary as the data comes in for detector monitoring and calibration. However, this usually only requires access to the most recent data. An alternative to storing the entire set in robotics would be to store the full DST and μDST sets along with a revolving set of about 3 months worth of RAW data. This would require an initial 40TB for the RAW data plus an additional 35TB/yr for the DST and μDST sets. The cost for such storage would be about $438K-2M for the first year and $205K-1.2M in subsequent years.

7.3 Bandwidth Requirements

During Run I DØ utilized two FDDI rings between the workstation clusters (one to the Alpha cluster, one to VMS/UNIX machines) and the DØ file server. The total utilized bandwidth on these two rings was typically in the 8-12MB/s range during their periods of heaviest use. In addition, local data access between workstations which did not transit the FDDI rings is estimated to have been in the 5MB/s range. Additional local I/O on workstations may have been in 10MB/s range. In total, DØ utilized around 25-30MB/s of bandwidth to data. If we wish to analyze ten times as much data in roughly the same amount of time, then we would conclude that we will need in the range of 250-300MB/s of bandwidth to data on the central systems. This should be viewed as a minimum requirement. We would likely benefit from any increased I/O capacity.

The required I/O capacity should be easily available on any SMP type system we are likely to use during Run II. I/O tests have been done on our existing SGI Challenge XL machines. We were able to realize 50MB/s of I/O to disk on a single IO4 card utilizing only 3 SCSI buses with Fast/Wide Elite-9 disk drives attached. Each Challenge machine is capable of supporting 4 IO4 cards with 8 SCSI buses each. Each IO4 card is rated at a maximum 320MB/s of throughput. Although backplane contention issues would prevent any single IO4 card from reaching its maximum rating, we can conclude that, even with existing Challenge technology which is near the end of its life cycle, we should have no difficulty attaining the required bandwidth to disk. This assumes that the central systems are not required to expend their resources serving large quantities of disk out to a distributed workstation cluster.

7.4 CPU Requirements

During Run I DØ used about 1500TINYs of CPU for analysis. This was distributed over the D0FS/FNALD0/D0TNG/D0SFT clusters. Scaling this number with expected Run II data size results in a minimum requirement of 15,000TINYs of compute capacity for analysis in Run II. The Run I usage is a number averaged over accounting periods of one week. Peak CPU demands sometimes resulted in long backlogs of jobs in batch queues, particularly in periods just before major conferences. When the large UNIX systems were first installed, the analyses that were run there were ones which simply could not be done on the VMS systems because of limited resources. These factors lead one to conclude that we were limited in Run I by the available CPU resources. If we restrict Run II resources to simply scale with the data then that situation will likely recur. We should strive to put at least 30,000-40,000TINYs of compute power online for Run II data analysis.

Effectively using 40,000TINYs of compute power with local data access will present a challenge to the experiment. SMP machines such as the Challenge XL are expandable to about

4000TINYs in a practical configuration if they are using the newest R10000 processors. That would require 10 such fully configured machines to provide the analysis computing. Any efficient use of such a system will require parallel processing and data access such as we now do with the Piaf system. The latest generation of SMP machines from SGI, the Origin series, is expandable to 128 processors and offers the possibility of providing 40,000TINYs on a tightly coupled system by the time Run II begins.

The current cost of a fully configured 128 processor Origin system is estimated to be in the $3-4M range. Historically large data handling CPU has improved by a factor of two in price/performance every 1.1 years. It should be reasonable to expect to phase a system of this size in over a two year period starting in '99 for around $750K/yr.

7.5 Spending Profiles

We will assume that the costs of farm systems and robotic storage are born by a common Computing Division purchase and do not come from funds directly allocated exclusively for DØ computing needs. In addition to the expenditures noted above we can expect to have on the order of $200K-300K/yr of miscellaneous expenses for such things as networking, video conferencing, and other peripherals.

The time scale for installing this equipment depends on the functionality it is to perform. The reconstruction farms must be in pace at the beginning of the run at near full capacity. This dictates that farm purchases must be made no later than fiscal year '99. In principle the full analysis CPU should be online near the beginning of the run, but this will simply not be feasible because of cost. It will be necessary to phase this in over a period of more than one year. All storage costs such as disk, serial media, and robotic capacity can be added in increments as needed.

In the table below we have assumed the lowest cost alternative in those cases where choices exist. In the case of serial media this means that we write only RAW, DST and μDST sets to tape, we use the lowest cost media (currently DLT 7000), and that the cost will have fallen by a factor of two by FY99. The cost for robotics may be unrealistically low since the lowest cost media is not associated with the lowest cost robotic and drive hardware. The farm expenditures listed are what we presume will be spent to satisfy DØ's needs alone, not the full lab expenditure to cover CDF as well.

Item	FY97	FY98	FY99	FY00
Farms	$250K	$500K	$500K	-
Robotics	-	-	$435K	$205K
Analysis CPU	-	-	$750K	$750K
Disk	-	-	$350K	$350K
Serial Media	-	-	$300K	$300K
Misc.	$200K	$200K	$300K	$300K

8 Summary

The DØ software and computing effort for Run II is now well on its way. At the time of the Software Workshop in November, 1995, we had the beginnings of a blueprint for the future, in the shape of the major recommendation to move to C++ and DSPACK, but no organization in place and no one working on Run II software (with the exception of UPG_GEANT and the prelimary versions of Run II tracking code, in Fortran) as a major project. At the time of the December, 1996, Workshop, we had

- The long term organization in place
- The recommendations, and some implementations, for software tools
- Subgroups in place for the critical Algorithms Group reconstruction effort
- A working prototype of the scintillating fiber tracking in C++
- A design and skeleton code for a C++ version of the CAJETS package
- Formal training in C++/OOAD for almost all the core people in the Run II reconstruction effort
- Working meetings of the Online software group
- A Run II code repository and prototype tools for making releases
- Joint CD/DØ/CDF groups in operation to tackle C++ issues, software enginnering tool choices, data management needs, and configuration management (with databases, we hope, soon to follow).
- A prototype of the Run II framework and program builder
- A prototype C++ interface to the object persistency mechanism
- 25 people actively involved in Run II software and computing, and the effort is still ramping up

There is a definite schedule for the reconstruction code and the online software, and an estimate of number of FTE'S required. Right now, we see a need to double the FTE effort involved with reconstruction over the course of the next year, and to triple the FTE effort in online software. Achieving these increases poses the major immediate challenge. The major long-term challenge is to live up to the difficult goal proposed in Section 1 – to build a system that is not only robust but flexible enough to respond to hardware and software changes that will improve our efficiency in producing physics from Run II.

References

[1] D. Adams *et al.*, "DØ Run II Computing Language and Event Data Persistency Recommendations," DØ Note 2819.

[2] http://www.fnal.gov/docs/working-groups/c++wg/

[3] http://www.fnal.gov/docs/working-groups/c++wg/interim.html

[4] http://www.fnal.gov/docs/working-groups/fpcltf/

[5] R.G.G. Cattell ed., "The Object Database Standard: ODMG-93, Release 1.2" Morgan Kaufmann, San Francisco, 1996.

[6] H. Greenlee *et al.*, http://www-d0.fnal.gov/~greenlee/dspack_odmg.ps.

[7] P. Bhat *et al.*, "DØ Database Working Group Report," DØ Note 3122.

[8] "Recommendation of the D0 Script Languages Committee"

[9] S. Fuess *et al.*, “D0 Analysis Computing Recommendations,” D0 Note 2537.

[10] A. Jonckheere, “Minimum Code Management Requirements,” http://www-d0.fnal.gov/software/tools/min_cmgt.txt, March 7, 1996.

[11] “Run 2 Code Management Report”, http://fnpspa.fnal.gov/Run2/document.html, May 12, 1996

[12] R. Harris ed., CDF Note CDF/DOC/COMP_UPG/PUBLIC/3891; D0 NOTE 3118; Computing Division Note GU0013; http://www-cdf.fnal.gov/offline/code_management/run2_cmgt.ps.

[13] A. Jonckheere, “Code Management, Getting Started at D0’, D0 Note: xxxx and http://d0wop.fnal.gov/software/tools/getting_started_at_d0.html, current draft Dec 3, 1996.

[14] “The BaBar Release Structure,” http://www.slac.standord.edu/BFROOT/dist/releases/current/SoftRelTools/SoftRelToolIntro.ps

[15] P. Bhat *et al.*, “Recommendations of the Software Tools Working Group — Code Development Environment,” D0 Note 2936.

[16] http://www.fnal.gov/cd/sweng/sweng.html

[17] D. Baden *et al.*, “Recommendations of the Software Tools Group Text Editing and Document/News Management,” D0 Note 2853.