11

Computing

There are a number of computing challenges for the proposed BaBar detector. The data rates are high for an e^+e^- collider, and there is a large quantity of useful data which must be stored and analyzed. Convenient interactive access must be available for the efficient extraction of physics results. In addition, the wide geographical distribution of the collaboration implies significant issues for remote access to the data and code, collaboration communication, and for remote computing.

The BaBar Computing System, as defined here, includes several aspects: (1) the online data acquisition, control, and monitoring hardware and software; (2) the offline computing hardware—CPUs ("farm" and desktop), storage, X servers, and networking—whether local to SLAC or at remote locations; (3) the software environment in which the computing work is done; and (4) the code itself used by the collaboration. All of these are important items for BaBar computing, although parts of these, including most of (2), are properly considered institutional infrastructure.

The next section reviews the computing requirements, and Section 11.2 gives an overview of the proposed computing system. Following these are more detailed discussions of the computing model, software environment, online system, reconstruction and analysis framework, computing support plan, integration issues, system responsibilities and management, and cost and schedule.

11.1 Requirements

In this section, the basic computing requirements for BaBar are summarized. These are divided into operational requirements and technical requirements. Operational requirements are broad capabilities that must be present in order for the system to respond adequately to the environment within which it must operate. Technical requirements are specific capabilities that the system must exhibit in order to meet the operational requirements, for example, the CPU power and bandwidth. Estimates are given for the early years of the experiment. It is recognized that the requirements may increase as the experiment progresses; hence, a general further requirement exists that the system be scalable to meet future needs.

11.1.1 Operational Requirements

The *BaBar* Computing System must support several activities which compete for available resources. These include program development by a distributed developer community, data acquisition, control and monitoring, bulk data reduction and Monte Carlo simulations, and interactive physics analysis.

As the collaboration is worldwide, a working scheme which in no way reduces the ability of the remote collaborators to contribute to the experiment must be implemented. The remote computing issues cover the way in which the collaborators exchange information (at the moment WWW, ftp, netnews, and mailing lists), how software development is integrated, and how to implement updates of software production releases, data analysis, Monte Carlo production, database access, and software licensing. Access to the data will be provided at regional centers, which will keep some terabytes of data.

The computing system must provide a real-time response adequate to perform its many roles. This impacts several areas. The most obvious is that of data acquisition. To benefit fully from the available luminosity at PEP-II, it is important to reduce as much as possible the effective deadtime for the online system. The time for routine operations that can interfere with data taking, such as run start and close-down or system reboot, should be small. Finally, the latency between the acquisition of the data and its analysis for physics results should be short compared to the duration of an accelerator scheduling cycle.

The computing system should display a uniform face to the users that hides any intrinsic heterogeneity and allows them to be productive at their home institutions, at regional centers, or at SLAC. It must be robust, both in the sense of exhibiting a high degree of reliability and high availability, and also in allowing short recovery time from a failure. This in turn implies a high degree of maintainability. Given the extended time over which the experiment will be active and the probability of unforeseen demands and changes in the requirements, the computing system must also be highly flexible and extensible, this being achieved with a minimum of effort and in time intervals dictated by the accelerator scheduling cycle. Finally, the computer system must balance ease of use with security so that developers can be productive, and also ensure that high quality, well-understood, and reproducible physics results can be achieved.

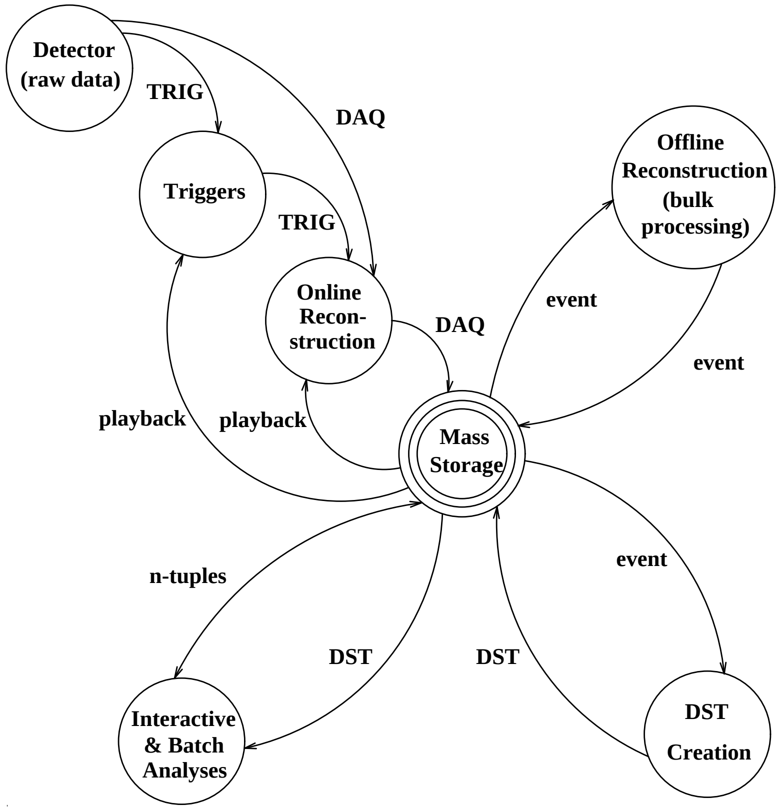

Figure 11-1. *Schematic of the data flow for the BaBar experiment.*

11.1.2 Technical Requirements

CPU Requirements

There are several substantial computing tasks that require significant CPU resources: the online system, including the Level 3 trigger and data acquisition, event sampling, detector monitoring, calibration, and control; offline reconstruction; Monte Carlo simulation; creation of DSTs from the reconstructed data, involving a reduction in event size and/or a reduction in the number of events; analysis of large DST datasets; and interactive analysis tasks. These tasks are represented in a data-flow diagram in Figure 11-1. The CPU power[1] required to handle this data flow is summarized in Table 11-1.

[1]The familiar term MIPS is used to describe CPU processing power. In terms of the current industry standard specifications, one MIPS represents a power corresponding to approximately one SPECint92 (integer performance) and one SPECfp92 (floating point performance).

Online (Level 3)	MIPS	3000
Reconstruction	MIPS·s/event events/year passes/year MIPS	25 10^9 2 5000
Monte Carlo	MIPS·s/event events/year MIPS	500 10^8 5000
DST Creation	MIPS	500
DST Analysis	MIPS	2000
Interactive Analysis	MIPS	5000

Table 11-1. *Estimated CPU power requirements.*

The online CPU requirement is estimated assuming a 2000 events/s input rate to the Level 3 farm. This rate is higher than the average anticipated rate, but is used to ensure that the system is able to keep up with the data flow under extreme conditions. This is quickly reduced to the order of 200 events/s using p_{t} and z information from the trigger. More complete event information is then used to reduce the rate to tape to 100 events/s. A full reconstruction will be done on approximately 30 events/s (of which ~10 events/s are hadronic via single photon) at $\mathcal{L} = 3 \times 10^{33}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$.

Based in part on the CLEO-II experience [CLE94], similar figures are derived for the CPU requirements relevant to event reconstruction and Monte Carlo simulation (5000 MIPS each). The CPU requirement for Monte Carlo simulation, as for the data acquisition and reconstruction, is based on a peak rate requirement, namely the ability to keep up with the peak hadronic event data rate. This corresponds to a rate of 10^8 Monte Carlo events per 10^7 seconds.

DST creation refers to a major pass through the data to create a selected dataset (in which the output event size may or may not be reduced. The generic term DST is used here to include differing definitions of DST, mini-DST, *etc.*). The CPU power required is computed according to:

$$\mathrm{MIPS} = 0.25(\mathrm{MIPS{\cdot}s/event}) \times 10^9(\mathrm{events}) \times 2(\mathrm{users})/10^6(\mathrm{s}),$$

which assumes that it is acceptable to have this task take less than a month, hence the 10^6 second time period; the two users allow for up to two such major passes through the data to be simultaneous. This estimate, of course, is a particular, and rather conventional, model for how an analysis progresses into selected subsets of the data. Alternatives will be investigated, such as using a direct access database technology [Bad90].

DST analysis is a substantial processing of a selected DST dataset for physics analysis. The formula used to estimate the CPU power in that case is

$$\mathrm{MIPS} = 1(\mathrm{MIPS{\cdot}s/event}) \times 10^8(\mathrm{events}) \times 20(\mathrm{users})/10^6(\mathrm{s}).$$

This assumes 20 such analyses occur once per year, each requiring one month of CPU time to process the full dataset of interest. This estimate is in addition to the resources required to design and debug such analyses.

Interactive analysis involves debugging, exploratory analysis, graphics, and program development. One hundred active users, each employing a 50 MIPS workstation, are anticipated.

In summary, if the numbers above are summed, the total CPU requirement is 20,000 MIPS. At least 10,000 MIPS have to be located at SLAC for data acquisition, reconstruction, and other tasks. However, the total installed MIPS may be lower depending on the model (Section 11.2.2); the requirements are based on keeping up with the data acquisition, effectively realtime, at the peak luminosity rate, rather than at the rate averaged over a year.

Throughput

Transformation of the raw data into the final reconstructed event information, and subsequent physics analysis, requires several data transfer steps (Figure 11-1). The most significant steps (in terms of required bandwidth) include:

- Transfer of data into the Level 3 online farm;
- Transfer of data from the Level 3 farm to mass storage (tape);
- Reading and writing the data in the bulk reconstruction;
- Writing of Monte Carlo events to tape;
- Reading through the data to create DSTs;
- Passes through the DSTs for physics analysis; and
- Interactive analysis on small datasets.

The estimated requirements are summarized in Table 11-2. The upper-limit requirement of 2000 events/s is assumed going into the Level 3 data acquisition farm, and the event size is estimated to be 25 kbytes [Por93], yielding a 50 Mbyte/s rate. The event size is assumed to double to 50 kbytes once the reconstruction information is added, but the rate decreases to 100 events/s. Hence, the reconstruction data rate is 7.5 Mbytes/s, 2.5 Mbytes/s input plus 5 Mbytes/s output. The Monte Carlo event size is also estimated at 50 kbytes, which, at the simulation rate of 10 events/s, implies a data rate of 0.5 Mbyte/s.

Task	Input (events/s)	Output (events/s)	Bandwidth (Mbytes/s)
DAQ to Level 3		2000	50
Level 3 to Tape		100	2.5
Reconstruction	100	100	7.5
Monte Carlo	10	10	0.5
DST Creation			$50(1+f)$
DST Analysis			5/task
Interactive Analysis			0.2/user
Wide Area			
$80\times$ Sessions			> 0.5
Data Transfer			0.5

Table 11-2. *Estimated bandwidth requirements. The f factor is explained in the text.*

DST creation and analysis represent a large network load. Substantial DST creation may occur simultaneously with the reconstruction via multiple output streams. However, additional passes through the data should be anticipated here. In DST creation, f is the product of the sample fraction times the average sampled event size divided by 50 kbytes. If f is small, and two such tasks are running, 100 Mbytes/s is required. The task of DST creation will typically be heavily I/O bound. DST analysis, with 20 analysis tasks simultaneously running, requires 100 Mbytes/s.

The interactive analysis number is estimated assuming a 2 kbytes/event reduced DST, and a throughput of 1000 events in ten seconds, appropriate for small datasets which are assumed to be analyzed at a desktop workstation. There may be an occasional need for interactive analyses on larger DSTs, with larger bandwidth requirements. It is anticipated that such analyses would take place on the same machines as the DST analysis tasks, supported with higher bandwidth.

Based on this analysis, aggregate network capacity for offline tasks in excess of 200 Mbytes/s is required, dominated by the reconstruction and the multiple reads through the data required for DST creation and analysis. However, the DST creation task, which is a potentially large burden specifically on tape access, may be greatly mitigated by having this process proceed largely via multiple output streams during reconstruction.

The wide-area network will be used for code development, interactive analysis, user communications, and transfer of small datasets (less than 10 Gbytes). A rule of thumb is that an X Window session requires 50 kbits/s. Thus, supporting an estimated 80 sessions nominally requires 0.5 Mbyte/s, except that the scaling is sublinear with number of sessions over a

Tape Storage:	Tbytes/yr
Raw Data	25
Reconstruction Output	50
Monte Carlo	5
DSTs	5

Disk Storage:	Tbytes/yr
DSTs	2
Databases	0.03

Table 11-3. *Estimated storage requirements.*

network. To transfer a small dataset in an acceptable time (a few hours) requires a similar bandwidth.

Storage

Table 11-3 summarizes the estimated storage requirements for the *BaBar* experiment. Approximately 100 Tbytes/yr of tape storage is required. If necessary, the 25 Tbytes of raw data could be moved offline after reconstruction. The 5 Tbytes/yr for DST storage is estimated by assuming 10 DSTs produced per year, with 10^7 events in each (sampling fraction of 0.01) and 50 kbytes/event.

Current experience at SLAC is that a (disk) staging space of 2% of the active tape storage is required. Alternatively, 2 Tbytes may be sufficient for 40% of the DSTs produced to be on disk. However, it is probably appropriate to regard this space as staging space, with the details of whether a particular job's data are already staged to disk transparent to the user. The database space requirement is implementation dependent but should be small compared with the event data requirement.

Disk space is also required for group code (and documentation, *etc.*) and for user space. The group code space will be relatively small, probably in the tens of gigabytes. The user space required will also be comparatively minor but cannot be neglected because of the large number of users. From developing experience, it appears that *at least* 100 Mbytes/user of disk space will be needed. If there are 300 users, this means at least 30 Gbytes.

11.2 Overview

This section gives a summary of the basic design choices which have been made and an overall description of the planned *BaBar* computing system.

11.2.1 Chosen Technologies

Choices for key computing technologies depend largely on capability and economics. Additional technology choices are then driven by those initial selections. Finally, some technologies are selected reflecting a judgment of how best to solve a particular problem. Given that the computing industry has become highly competitive and that technologies will evolve quickly on the timescale of the *BaBar* experiment, it is important to identify available and flexible solutions whenever possible. Thus, some of the initial selections may be considered working decisions; the initial design is expected to evolve between this design and the final implementation and final uses (a period of approximately 15 years).

The magnitude of the computing need overwhelms the traditional mainframe approach and dictates the use of a distributed, multiprocessor environment. In the following discussion, it is assumed that all computing, both online and offline, will share a common environment. It is recognized that to accommodate the diversity within a widespread collaboration and changes occurring within the computing marketplace, special care must be taken to plan for multiple platform support. This is both a hardware and a software issue; thus, generic tools which are platform independent have been and continue to be sought. At present, this implies the use of workstations utilizing inexpensive (RISC) technology and operating system software within the UNIX family. While it is too early to decide upon a specific networking medium, the industry-standard Transmission Control Protocol/Internet Protocol (TCP/IP) is expected to survive and will, therefore, play a major role both in local-area and wide-area network communications. The use of an object-oriented (OO) approach will be encouraged for all software development within the collaboration. The primary programming language will be C++. Fortran 90 will be an acceptable alternative. Windowing technology will be the X11 client-server system with the Motif library and window manager. Printing will support both text and the PostScript page description language. The management of program code releases will be, in part, based upon the Concurrent Versions System (CVS). World Wide Web (WWW) will serve as the primary mechanism for information access. The online system will be built around the Experimental Physics and Industrial Control System (EPICS) (Section 11.5.5). This, in turn, relies upon the VxWorks real-time executive; although it is expected that this requirement will be relaxed within a few years as EPICS becomes POSIX[2] compliant.

11.2.2 Overall Description

Computing Model

The *BaBar* computing model is based on systems of Unix processors. Detailed choices will be deferred until need actually arises in order to take advantage of the rapid developments in computing technology. With the technology of today, a satisfactory solution can be built with a combination of Ethernet and FDDI local networking with nonblocking Ethernet-FDDI hubs and FDDI switches, SCSI-2 disk farms, and helical-scan tape technology in existing tape silos. A small number of remote computing centers, based on similar technology choices, are anticipated, with wide-area communications achieved by a combination of Internet connections and 3490 silo cartridge or other tape (*e.g.,* 8 mm) transfers.

Software Environment

A consistent, robust, and easy-to-maintain software environment will be created by resorting to common facilities for writing, managing, and distributing code relevant to all the computing tasks. Recent advances in software engineering, such as the object-oriented methodology and programming languages, will be exploited. An approach has been chosen which includes the rapid creation of guidelines and standards documents, code templates, and sample programs, together with the establishment of a code development environment that supports the needs of the software developers while also ensuring that the end users' need for stability is also addressed.

Online System

The online system has not only to support the steady state operation of the detector for physics data acquisition, but also has to deal with the conflicting demands of detector commissioning, calibration, and diagnostics. A system that accommodates these demands is described in terms of multiple viewpoints, from the most abstract which deals with the need to control multiple logical experiments or "partitions" during commissioning, to the most concrete which deals with the detailed control protocols among modules within the data acquisition system.

[2]IEEE Portable Operating System Interface for Computing Environments.

Reconstruction and Analysis Framework

A common framework for applications is being developed that will allow physicist-developed code to be used for online triggering and monitoring, for bulk reconstruction and Monte Carlo simulations, and for physics analysis. This framework will shield the application programs from the different operating conditions in these areas and allow single applications to operate in all areas, accepting event data from real-time event servers in the online area, or from disk or tape files offline, and supporting output of analyzed data to a variety of output destinations including the real-time event server and disk or tape files. This framework must furthermore provide for both interactive and batch styles of operation and accommodate the conflicting needs of experts and novices. Both text-based and graphics-based user interfaces are under development in order to accommodate these needs, addressing also the needs for bulk job submission and bookkeeping.

Computing Support

Computing support refers to both internal (within the collaboration) and external (university and laboratory support organizations) activities. Internally, this support will be coordinated directly by the computing system group. Designated members (Code Coordinators) of the collaboration will be assigned to look after the various software packages, to manage the data production and distribution tasks, to maintain the different databases, and to provide system management support for the online computers. The collaboration will see that appropriate documentation is generated, training needs are identified and covered, and that specific tools and utilities for the experiment are written. These support roles are expected to consume a significant amount of manpower within the collaboration.

External support will be provided by support staff at the collaborating institutions and is expected to include a wide variety of services. The overall computing system must be designed with the active participation and cooperation of these groups. Hardware elements must be planned, purchased, installed, operated, and maintained. This includes networking, mass storage, and CPUs. These heterogeneous and distributed computer systems must be managed with respect to user accounts, disk and other resources, backups, system tuning, *etc.* Wide-area networking operations will require continual monitoring with occasional attention. Software must be licensed, installed, and maintained. The details of data storage and access must be designed and implemented. A system for efficiently utilizing distributed computing cycles must be installed and maintained. Finally, the support staff must spend some fraction of their time in system analysis, technology tracking, and strategic planning.

11.3 Computing Model

A comprehensive computing model must satisfy the functional requirements outlined in Section 11.1. In addition, such a model must address practical considerations, for example: the "abilities" (usability, flexibility, scalability, reliability, maintainability, and affordability); smooth integration of all hardware and software components; ability to evolve in response to commercial developments; and compatibility of SLAC on-site computing with that of off-site collaborators. In the sections that follow, an architecture which addresses these requirements, a model to describe its overall operation, and a cost model based upon present and near-future technologies are presented.

11.3.1 Architectural Model

The overall computing system architecture includes a number of basic hardware and software components. Necessary hardware components include the elements of processing power (*e.g.,* computers), networking, and mass storage. The main software component is the operating system. A schematic model containing these components is given in Figure 11-2.

CPU Complex

The *BaBar* Collaboration has concluded that, currently, the most cost-effective way to satisfy the basic CPU requirements is through the use of multiple microprocessor-based computers. Microprocessors utilizing a RISC architecture, as found in Unix workstations, represent the optimum in today's market. A continued high rate of growth in computing power of this kind is expected for at least the next five years. Alternative computing architectures are not excluded in this model, and the possibility of a transition sometime during the period of this experiment has been considered.

A commitment to clusters of distributed workstations for an online system, reconstruction, Monte Carlo simulation, and interactive data analysis, along with general software development and debugging, requires a sophisticated and mature, multitasking, networked operating system. This requirement is met by using Unix, offered by all major workstation vendors. Such a solution also tends to capitalize on relatively high-volume, inexpensive, components (*e.g.,* video monitors, SCSI disks, and RAM), thus offering a degree of vendor independence both for hardware and software.

Commercial development of workstation-based computing is proceeding in several directions. For example, workstation clusters currently tend to use single-processor implementations.

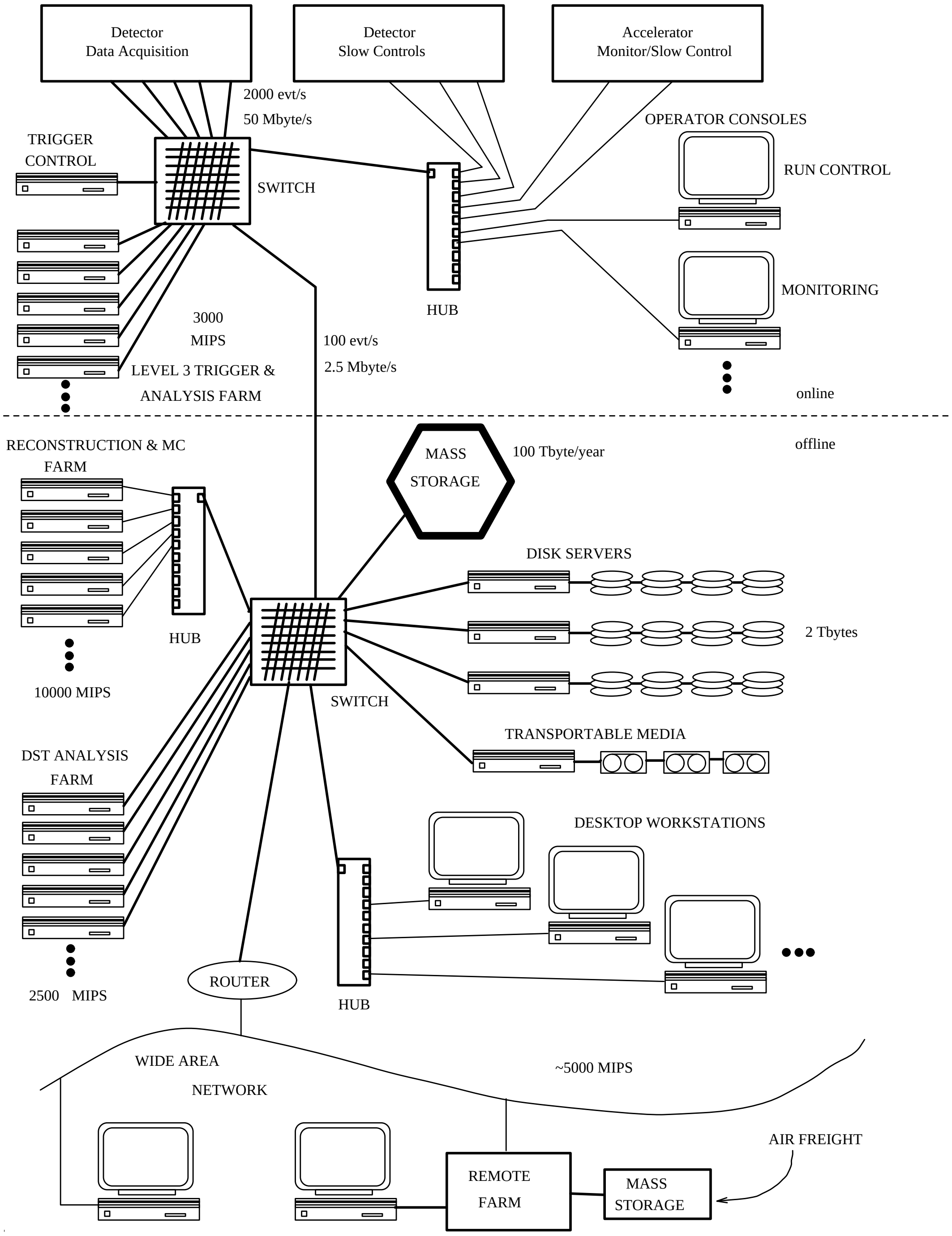

Figure 11-2. *Conceptual computing architecture.*

There is a trend towards multiprocessor cluster implementations that improves processing performance by virtue of a high-speed backplane interconnecting the individual CPUs. Current examples of this architecture include the SGI Power Challenge, the IBM SP-2, and the Convex Exemplar. Such multiprocessor implementations could be used to optimize the design, but only if the costs become competitive.

The UNIX operating system is also undergoing a series of changes. The direction of change appears to be toward standardization of both the basic command set and system libraries. By carefully adhering to such standards (*e.g.,* POSIX), one can attempt to minimize the work required to make a transition to another computer environment should that become necessary. Given the rapid pace of developments within the computing industry, the effort to abide by such standards within the collaboration is deemed essential.

Networking

There are two levels at which networking is crucial to this experiment: local networking at SLAC, because of the large number of machines and high data rates involved both online and offline; and wide area networking, due to the large, widespread international collaboration. Local area networking refers to the links between the various components of the experiment, including those to desktop machines and the central computing complex. Wide area networking refers to those links between the various collaborating institutions and to SLAC. While the SLAC LAN is completely under lab control, the WAN is not. *BaBar* may hope to influence various funding agencies to accommodate WAN needs but cannot expect to sponsor significant upgrades due to the high costs, as shall be discussed later.

A networking system consists of various components and subsystems, including: workstation and other device interfaces; switches and hubs; various diagnostic devices; and the cable plant. The basic LAN architecture consists of a switched network of point-to-point links. In some cases, the I/O demands on a particular system may allow point-to-point links to be replaced by a short series of daisy-chained devices, or for a relatively expensive technology to be replaced by a cheaper (and lower performance) technology. The latter concept is illustrated in Figure 11-2 in the distinction between the Reconstruction & MC farm, characterized by a relatively high CPU:I/O ratio, and the DST Analysis farm, in which condensed data is read at a high rate with relatively little processing per event. The networking protocol will be TCP/IP, an industry standard available on all UNIX machines.

Mass Storage

A large amount of reliable, inexpensive storage with fast, automated access from the UNIX environment is needed. Mass storage technologies, like those in other areas of computing, are rapidly developing within the industry. Current mass storage candidates are typically expensive and unique to a particular vendor. While it seems unlikely that on *BaBar*'s timescale a *de facto* industry standard will emerge, several near-future products are expected to address these needs satisfactorily. Unlike the computing and network environments, however, the mass storage system is likely to be rather specialized.

The transfer of data between SLAC and remote collaborator sites may be complicated if the supported media at the various institutions are incompatible. For transfers to such institutions, an affordable and transportable data medium will be essential. Currently, 8 mm Exabyte tape cartridges are supported at SLAC and are in widespread use, although alternative formats (*e.g.,* 4 mm and DLT) are gaining in popularity. In general, capacity and performance are increasing while the cost per stored byte is decreasing. Hence, the decision as to which technology to use will be postponed until it is needed.

Prototype Compute Farm

SLAC has been pursuing R&D towards this model in the form of a prototype compute farm, shown in Figure 11-3, designed to exercise these ideas. Two batch environments are available for testing: LoadLeveler from IBM and Load Sharing Facility from Platform Computing. SLAC Computing Services (SCS) is working with CERN on the latest data staging software being developed as part of CERN's SHIFT. It is anticipated that in the next several years, there will be investigations into commercial products which support the IEEE Mass Storage Model to provide data staging and hierarchical storage management.

11.3.2 Operational Model

Hierarchy

A hierarchy of sites exists in the computing for the *BaBar* experiment. The data will be recorded at SLAC, giving SLAC a unique position. Demands of detector debugging and monitoring, physics analysis by collaborators resident at SLAC, and simplifications in data handling will mean that much, if not all, of the bulk data processing will be performed at SLAC. Next, there exists, within the collaboration several sites with the potential to provide computing facilities (data storage, computing capacity, and support) similar to those at SLAC. These regional centers will serve as analysis centers, providing convenient data access to institutes with poor network connections to SLAC and to those which cannot provide the levels of support required to maintain copies of the data locally. These centers could also be used for data processing, DST production, *etc.*, as performed at SLAC. Finally, there are the computing facilities in all of the other collaborating institutions. This final grouping

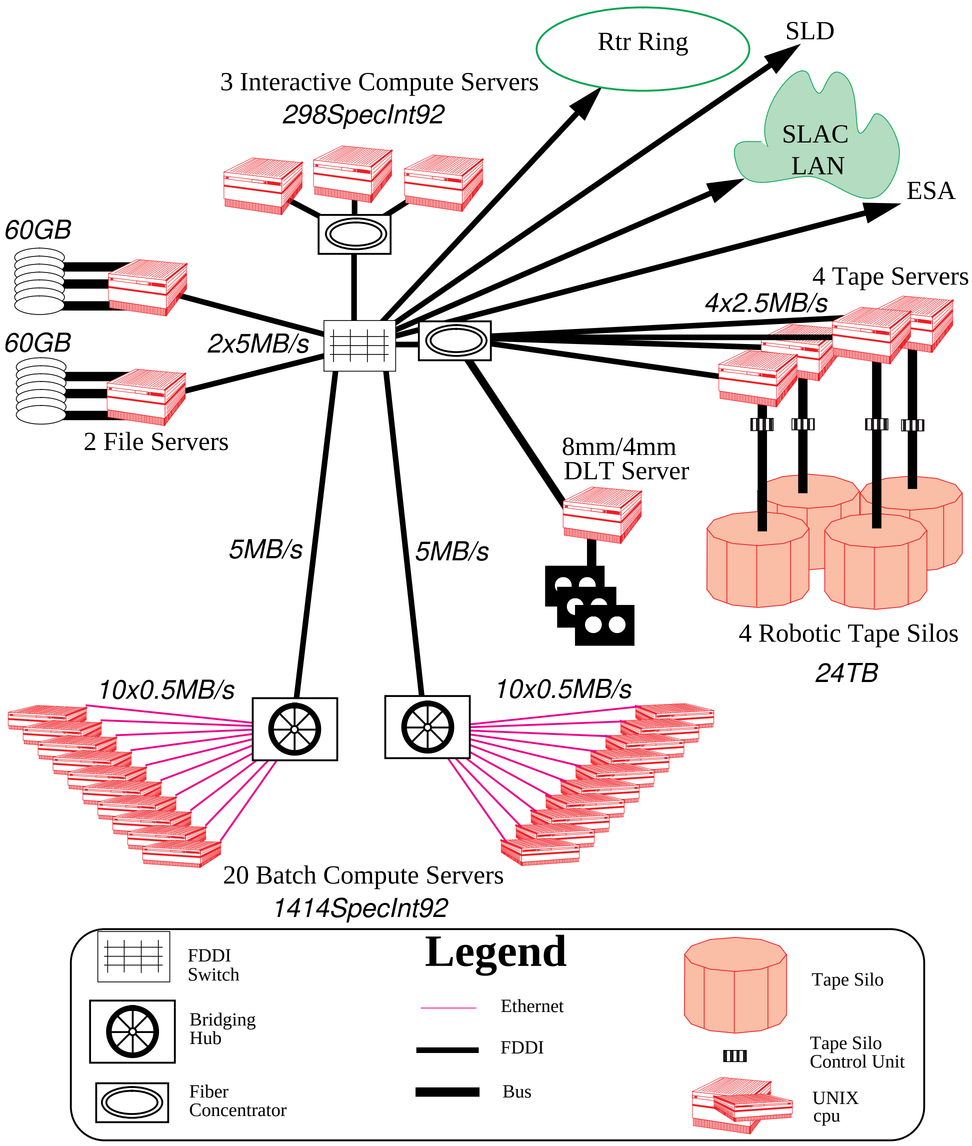

Figure 11-3. *Prototype* UNIX *compute farm at SLAC.*

of home institutions obviously includes a wide range of capabilities. They are characterized here, however, by the fact that they will not be used to perform bulk data processing, and are unlikely to want copies of significant fractions of the raw or processed data. It is envisaged that data analysis in the home institutions will be based (in terms of access to bulk data) around the data store at SLAC or a regional center, whichever is most convenient. Home institutions may want copies of DSTs. These again could come from SLAC or a regional center. It is quite possible that some of these institutions will contribute to Monte Carlo generation.

SLAC Site

It is essential to have only one master copy of production software and databases. The SLAC site is an obvious choice for these, especially during the installation and early running of the experiment when updates and changes may be required frequently and quickly. This model of organization has already been implemented by *BaBar* for software development. The use of CVS/rCVS places central control of the master copies of all code at SLAC while allowing users at remote sites access to this code with a mechanism for including changes back into the master. The user's view of this process is essentially the same whether they are based at SLAC or at a collaborating institution.

The benefits of performing bulk processing at the site of the experiment have been proven many times. Most large experiments now do this task routinely (ALEPH, DELPHI, OPAL, L3, H1, ZEUS, CDF, D0). Though other models do exist (E791), the quasi-online processing possible with local facilities is an efficient use of resources and has significant advantages in producing results more quickly. Clearly, the ability to perform bulk data processing is required at SLAC. A batch system is needed to facilitate this processing.

The presence of the raw and processed data, as well as subsystem experts, at SLAC makes the SLAC site an obvious place for data analysis. To make the most effective use of this requires that an efficient analysis framework be maintained at SLAC, in particular, one which allows users not actually based at SLAC simple and efficient access. This system should not require that remote users log on to the SLAC computers. Remote users should have effective access to SLAC's batch system allowing creation of small DSTs or n-tuples which can easily be transferred back to the home institution for interactive analysis. Current WAN technology allows n-tuples of many tens of megabytes to be efficiently transferred in this way. For larger datasets, high-density tape technology provides a method of transport. The ability to ship data in this way requires additional copying facilities at SLAC.

Regional Centers

Support for regional centers has significant manpower and cost implications both at the regional centers and at SLAC. However, there are distinct advantages to having access to a local data store. The regional centers will serve as data store, processing, and analysis centers. A *complete* data sample and support structure could be maintained at these sites. Regional centers will help affiliated remote users by providing easy access to the data and enabling them to build up the strength of a community.

The concentration of computing and data handling capabilities found at regional computing centers also makes them suitable for large scale Monte Carlo productions. Experience has shown that this is an effective use of these centers and an efficient use of collaboration resources. This also fits well with the regional center's role in general computing support for smaller collaborating institutions.

Regional centers already act as concentrations of support effort in *BaBar*. For example, rather than many institutions independently attempting to keep all software and libraries up to date over poor network connections, this effort is concentrated in a few places. Other institutions then make use of this local infrastructure directly via NFS/AFS and file transfer using the much better network connectivity that they have to the regional centers.

There will be varying forms of regional centers defined by the actual level of service and support they provide. This will be influenced by many factors. International networking and the costs of data transport may change significantly over the next five to ten years. The regional centers could play a role analogous to SLAC or any other home institution, or, more likely, somewhere in between.

Two sites have already expressed strong interest in the Regional Center model; CCIN2P3 in France and DRAL in the UK. Possible further sites in Europe or North America are not excluded, but the cost and effort associated with the copying of large quantities of data are likely to be such that the number of such centers will always be limited. Both CCIN2P3 and DRAL already provide large scale computing facilities for HEP experiments (including large UNIX workstation farms) on a similar scale to SLAC and are equipped with large tape robot systems compatible with the one at SLAC. Both the UK and France already make use of good national wide area networks, with *BaBar* software being maintained centrally and accessed by collaborating institutions via the network/NFS.

Home Institutions

Physicists in the home institutions should be able to contribute to code development remotely. The mechanisms for this are already in place. The long term success requires that the collaboration computing model remains broad based in terms of supported hardware, application of coding standards, and support for remote access and transport of small datasets.

It is anticipated that the analysis work of remote users will be centered at SLAC or a regional center in terms of access to the bulk data or full DST. Physicists will then transfer n-tuples or small DSTs to their own institutions for intensive interactive work.

Many home institutions have significant computing capability, and we may wish to make use of this for Monte Carlo generation, for example. The ability to generate Monte Carlo or reprocess small data samples locally, will almost certainly be required by many physicists.

Code, Data, and Database Access

The access to code and data is covered elsewhere in this chapter. Providing convenient access to code for remote sites also has repercussions, however. For example, facilitating the running of *BaBar* code on essentially any home institution (UNIX) computer entails the risk of potential problems with compatibility of results from different machines, architectures, or flavors of operating system. This problem could become particularly important when considering reprocessing the data at one or more regional centers or home institutions. The collaboration is aware of these problems and aims to organize the production and analysis efforts so as to minimize their effects.

Implicit in having good access to the software and data samples is the requirement of access to the collaboration databases. Any institution processing or reprocessing data, generating Monte Carlo, monitoring detector performance, *etc.*, will require access to one or more databases. This access could be achieved by real-time queries to a master database at SLAC, over the WAN. This seems impractical today, especially when good real-time performance is required, for example, during a batch job. Most likely, copies of all or part of the database must be made available to collaborating institutions. These copies will almost certainly require frequent, automatic updates. The ability to do this will be important when choosing a suitable database technology for the experiment.

License Issues

It is likely that at least some of the software used by the *BaBar* collaboration will be licensed commercially. Given that some of this will be duplicated in collaborating institutions, the collaboration should try to enter into blanket license agreements, covering SLAC and remote sites, in order to reduce the total cost of these licenses to the collaboration. For example, the possibility of purchasing AFS client licenses which are extensions of the Stanford AFS server license, as CERN has done with theirs, will be investigated.

11.3.3 Baseline Cost Model

It is too early to purchase the bulk of computing equipment needed for *BaBar* for many reasons: it is not needed immediately; decisions about inter-institution compatibility have not yet been made; technology is changing rapidly; and costs are decreasing. However, it is both useful (as a reality check) and necessary (to provide cost estimates for the funding agencies) to detail the design of a computing system using current and near-future technologies. In this section, such a system following the basic architectural plan set forth earlier (Figure 11-2) is described.

Networking

The baseline design specifies components available today (December 1994). The expectation is that the actual implementation will evolve with experience and product availability as items are required. The baseline design of the network should allow relatively easy migration to emerging technologies as they become cost effective. Based on the data flow, the various network links in the schematic diagram are addressed below.

Online Networking

The online networking is used for data acquisition, trigger, and online monitor/control. It is based on point-to-point FDDI links connecting the Level 1 single board computers to a cross-bar switch (Digital's GIGAswitch [GIG93]) and finally to the Level 3 CPU farm. A single GIGAswitch will provide 16 connections to the FDDI rings containing the front-end data acquisition crates; direct connections to 14 Level 3 farm CPUs; a link to a local Ethernet hub; and a single long-haul link from the IR hall to the computer center's mass storage silo. All other online networking for program development and monitor/control will use lower cost Ethernet links.

Local Area Networking

The design for local area networking is based on switched 10 Mbits/s Ethernet and 100 Mbits/s FDDI technologies. In particular, Digital's GIGAswitch and Alantec's Powerhub 3000 Ethernet switch [HUB94] are used to interconnect workstation farms, mass storage servers, and desktop machines as described below. The link between online and offline Gigaswitches will be an FDDI link running over existing multimode fiber between IR-2 and the computer center, a distance of about 5000 ft.

The need to follow activities in ATM (Asynchronous Transfer Mode) and the 100 Mbits/s Ethernet switching and interface markets as they evolve, is recognized. To gain practical experience and skills, it will be necessary to make serious on-site pilot studies of ATM and 100 Mbits/s Ethernet in 1995 or early 1996.

Wide Area Networking

All of the computing tasks (other than data acquisition) can occur at the sites of remote collaborators. The degree to which this may be done is regulated by the computing investment at the remote sites and by the development of the WAN connectivity between SLAC and the collaborating institutions in the following years.

Today, SLAC has two T1 (1.544 Mbits/s) connections to ESnet, one to Caltech, the second to LBL. In addition, there is a 10 Mbits/s Ethernet microwave link to Stanford University that is used to access Stanford and to provide backup access to the rest of the world if the ESnet links are down. Wide area networking support to first order is outsourced to ESnet and Stanford University.

Data rates and response times between SLAC and other sites depend on the distance, the number of network hops, the speed of the links between hops on the route between SLAC and the other site, implementation details such as window and packet sizes, the error rates, and how busy the link is. Typical values for US sites vary from 90 kbytes/s and 12 ms ping response time, to 12 kbytes/s and 120 ms. For European nodes, the numbers are 10 kbytes/s and 200–360 ms.

As a minimum, all collaborators will need reasonable WAN connectivity to SLAC for the purposes of transferring code and performing interactive work using the X Window system. For X sessions, one study [Aba94] shows that the number of concurrent X sessions between adjacent T1 ESnet nodes is on the order of 24. Other studies [NCD89] indicate that the load imposed by an X terminal is approximately 50 kbits/s.

In spring of 1995, SLAC plans to upgrade its ESnet links to T3 links (45 Mbits/s). Recent measurements made by ESnet [Bos94] for one hop on unloaded T3 links between LLNL, LANL, and PPNL gave TCP data rates of between 0.58 and 1.72 Mbytes/s for optimized settings of window, packet, and internal write buffer sizes. Depending on budgets and demand, further upgrades of ESnet links to OC3 speeds (155 Mbits/s) may be initiated in 1995 for one or two sites, followed by other major sites a year later. The next step after OC3 is OC12 at 622 Mbits/s. Technically, upgrading to such links should be feasible in 1996/1997.

International links are more expensive than mainland US links. Today's cost for the ESnet T1 (1.54 Mbits/s) link from Princeton to Germany is about $12K/month for the US half circuit. This is one-fifth of the total cost of the link. One might expect a T3 link to be four to six times as expensive [Bos94].

Connectivity between the wide area network and the on-site network will be provided through a firewall FDDI router with connections to the SLAC FDDI backbone and the FDDI ring to which the off-site routers will be attached.

Compute Farms and Other Processors

Online System. Software development, run control, and monitoring tasks will use general purpose workstations interconnected with each other and with the computer center as described above. The Level 3 farm, which functions as a part of the data trigger, online reconstruction, event tagging, and event monitoring, will also use general purpose workstations.

Reconstruction and Monte Carlo Farm. Event reconstruction is done in the offline reconstruction and Monte Carlo farm. Data from tape are first staged to a series of disk servers via FDDI connections through the offline switch. Each reconstruction farm machine is directly connected to a network hub, which in turn connects to the disk server via the switch. Both reconstruction and Monte Carlo tasks tend to be CPU bound rather than I/O bound, thus allowing the use of slower and less-expensive Ethernet network links to these machines.

The data aggregate rates for this farm are 8 Mbytes/s (Table 11-2). Assuming the nodes in the farm are rated at an average of 200 MIPS each, there will be about 50 nodes to provide the 10,000 MIPS required (Table 11-1). Using existing technology, the networking requirements can be met by connecting the nodes to two Alantec Powerhub 3000s. Each of the Powerhubs will support 12 switched full speed (10 Mbits/s) Ethernet connections and an FDDI connection. Each Ethernet connection will have two to three nodes attached via Unshielded Twisted Pair (UTP) connectors, for up to 36 nodes per Powerhub. Each of the two FDDI connections (one for each Powerhub) will attach to an FDDI port on a GIGAswitch.

The DST Analysis Farm. Creation and analysis of DSTs occur in a second farm of machines with direct FDDI connections to the offline switch. These jobs are I/O intensive, thus requiring the higher performance and more costly FDDI network links. Data flow is similar to that for the reconstruction and Monte Carlo, from the mass storage to disk servers and finally to/from the DST farm.

The aggregate network capacity required for the DST analysis farm, as estimated in the preceding requirements section, is 200 Mbytes/s. The compute requirements (Table 11-1) of 2500 MIPS could be met with a farm of thirteen 200 MIPS processors. To handle the bandwidth requirement of 200 Mbytes/s, however, about 20 nodes will be needed, assuming each node can support a maximum of 11 Mbytes/s FDDI transfer rate. In this model, each processor node will have its own dedicated FDDI port on an offline GIGAswitch. Measurements must be made to see what FDDI transfer rates can be supported by these nodes while still allowing them to deliver the requisite compute cycles for DST creation and analysis. One estimate for an ATM switching network [Dou94] indicates that approximately 8 MIPS of CPU power are required to deliver each megabyte per second of data. If this turns out to be a limitation, then an alternative is to double the number of farm nodes (hence, halving the required bandwidth for each node) and for pairs of nodes to share a single GIGAswitch port.

Desktop Workstations. Interactive analysis involves the manipulation (and sometimes creation) of n-tuple datasets, and other event data in the highly interactive manner of PAW or HippoDraw. This is also an I/O intensive task, but in general, it involves data sets of modest size. This type of exploratory analysis occurs on desktop workstations. The data flow is identical to that of the farm machines.

The desktop workstations are expected to be mainly located in the Central Lab, the Central Lab Annex, the Warehouse, IR-2, and the new B Factory building. The data rates for individual workstations are 0.2 Mbyte/s (Table 11-2). This is well within the capability of the 10 Mbits/s Ethernet interfaces of today's workstations. To ensure that multiple workstations do not require more aggregate bandwidth than can be sustained on a single 10 Mbits/s Ethernet segment, each segment will support only a few workstations. As more bandwidth is required for a particular desktop, the number of desktops on a segment can be reduced to one. The individual segments will be connected to an Ethernet switching hub located in the same building as the desktop, each with up to 36 nodes attached. In buildings with large (> 30) concentrations of workstations, multiple switching hubs will be deployed, each hub being located close to a workgroup, and in turn, connected via a multimode fiber to a FDDI port on the offline GIGAswitch.

Mass Storage

Tape Mass Storage System. A relatively small number of candidate mass storage technologies available today might serve *BaBar*'s needs. One seems especially well suited for use at the SLAC site, where four STK robotic silos with 3490 tape technology are currently in use. Each tape cartridge in this system holds 800 Mbytes (before hardware compression), while each tape controller (connected to four tape drives) has a maximum throughput of about 2.5 Mbytes/s. The total storage capacity is thus 24 Tbytes. An attractive upgrade option involves migrating to STK's Redwood helical scan technology. The expectation is that this will initially provide 25 Gbytes of data per cartridge, about 15 Mbytes/s maximum throughput per tape drive/controller pair for a total storage capacity of 600 Tbytes. Cartridge capacity is expected to quickly rise to 40 Gbytes and eventually to 80 Gbytes. Such a system satisfies the *BaBar* storage requirements while reusing a significant amount of existing equipment. This solution may also be attractive to remote regional computing centers within the collaboration where there is already a large investment in STK silo equipment.

Mass Storage Server. The data rate from the online to the mass storage is 2.5 Mbytes/s. In addition, there will be 7.5 Mbytes/s from the reconstruction and 0.5 Mbyte/s from the Monte Carlo generation to tape (Table 11-2). To this must be added the requirements for staging analyzed data from tape to disk. This requirement is assumed to add an extra 5 Mbytes/s which would be sufficient to allow the 2 Tbytes in the disk farm to be staged from mass storage twice per week. The aggregate data transfer rates for the mass storage are thus about 15.5 Mbytes/s. To this must be added the requirements for the creation of major DSTs, which is forseen to take place mainly during the reconstruction. To the extent that additional major passes must be accomplished in short time periods, the model may require scaling to additional bandwidth.

The baseline design calls for the mass storage to be served by four nodes. Each of these nodes will have its own dedicated FDDI connection to the offline GIGAswitch. The transfer rates achievable on the FDDI are on the order of 11 Mbytes/s [Che93]. Initially, the performance is expected to be limited by the workstation servers and the interfaces.

Disk Mass Storage System. The disk servers have bandwidth requirements similar to the DST analysis farm since that is where most of the data are needed. Using the GIGAswitch model, a need of about 20 disk server nodes each with a dedicated FDDI connection to the GIGAswitch is again anticipated. Today's model of GIGAswitch can only accommodate 32 FDDI connections, so at least two GIGAswitches will be needed to support the DST analysis farm and the disk servers. This will require segmenting the DST analysis farm and disk servers, each segment having connections to the reconstruction farm, the mass storage, the desktop workstations, the transportable media, and the external router. The connections between the two segments will be made by one or two FDDI links. This segmentation will need further evaluation to ensure a reasonable balance can be achieved. A new model of GIGAswitch is expected to have 52 ports with 155 Mbits/s ATM, and will be evaluated as part of the on-going design study.

Transportable Media. Standard 8 mm helical scan magnetic tapes will serve as the basis for data transfer to institutions without STK silo compatible drives. The data rate requirements for the transportable media are expected to be modest and easily handled by a single FDDI connection from the server to a GIGAswitch. If more than one server is required, then the servers will share a single FDDI port on the GIGAswitch.

The decision to use 8 mm tape in the baseline design stems from its widespread use within the physics community and current support at SLAC. Other technologies, such as 4 mm or DLT, are also possible for comparable costs. However, at most, one medium besides the silo medium would be supported by the collaboration.

11.4 Software Environment

The basic requirements for the software infrastructure are that it provide an environment in which developers are highly productive, provide the necessary stability to ensure that high-quality, reproducible physics results may be achieved, and also be flexible so that changes in both the detector hardware and physics goals may be accommodated quickly and efficiently. Applying the principles of commonality across all areas of software development, from the real-time systems within the data acquisition and trigger systems through the online and monitoring systems, bulk data reduction and Monte Carlo generation, and finally physics analysis, can bring about important benefits. However, application of these principles must not be at the expense of productivity and must take into account that there are different demands in the various application areas.

11.4.1 Software Methodologies and Languages

Types of Developers

Several types of software development exist within the collaboration, each type having different requirements in its support levels. Individual physicists developing their own analysis programs should not be overburdened with rules and regulations, whereas software developed in the real-time environment of the data acquisition and trigger systems must be subject to rigorous quality tests to ensure that physics data are not lost. Intermediate between these extremes is the software for the bulk reduction of the raw data and the online and monitoring software that ensures that the data from the experiment are of high quality. Within each of these levels, an appropriate level of quality control will be applied as described later.

Software Methodology

Our overall approach to software development is based upon the object-oriented (OO) methodology. This methodology allows for a direct, intuitive association of physical entities (*e.g.,* detectors and particles) and abstract entities (*e.g.,* detector status and run control) with software objects. Commonalities between objects are also identified and exploited via the concepts of inheritance and overloading. Mapping of software objects into lines of code is most naturally done with an object-oriented programming language such as C++. The debugging and maintenance phases of development are well supported by the modularity imposed by the use of software objects. Many aspects of the object-oriented methodology can be automated through the use of computer aided software engineering (CASE) tools.

Several particular OO methodologies and notations have been investigated, including Booch, Schlaer-Mellor, OMT, and Objecteering [OOM94]. Two CASE tools, one supporting the Booch methodology, the other supporting the Objecteering methodology, are under detailed consideration. Each provides automatic C++ code generation as well as some reverse engineering capabilities for existing C++ code.

Programming Languages

After much consideration and experimentation, it has been decided to base most of the software development on the C++ programming language while continuing to support the Fortran 90 language. Most of the real-time and online systems, together with the core of the offline system, will thus be implemented in C++, while adequate functionality will still be provided for the Fortran 90 user. Such an approach is believed to produce the most robust applications, taking advantage both of the power of the OO approach and C++ and of the large existing Fortran code base and many man-years of experience with Fortran. This approach implies a mixed language environment and therefore carries the burden of additional support, but it is believed to be the most realistic approach, given the schedule demands and learning curve of the new OO paradigm.

Cross-Language Issues

Efficient access to software modules and data from both of the supported programming languages is obviously of crucial importance. Significant R&D has been performed in understanding how to accomplish this. Of central importance is the fact that the client programmer's view of the data format is established not by the data itself, which might be highly compressed in order to minimize the disk or tape space requirements, but by the interface that the corresponding object or module provides. Transforming the data between its compressed, internal form, to that seen by the user programmer, can be achieved in an efficient manner for both C++ and Fortran 90. Further R&D is underway to see whether both representations can be derived from a common source. The Interface Definition Language (IDL), a component of the Common Object Request Broker Architecture (CORBA) [OMG91], allows this and has the additional advantage that it supports distributed object applications. An incomplete prototype IDL compiler that generates both C++ and Fortran 90 stubs and skeletons has been demonstrated and shows some promise, but more work in a real application environment will be necessary in order to decide whether to continue this project. This approach seems to be well matched to fairly coarse-grained clients and servers, such as the packages described in the code management section.

11.4.2 Development Environment

Approach

In recommending that most program development be performed using the OO paradigm and implemented using C++, it is recognized that ease of use will have to be maintained, and therefore, support tools will be necessary in order to minimize the learning curve. The approach is aimed at establishing an environment in which newcomers can quickly be productive. Tools that are already in place, or shortly will be established, include code guidelines, code templates, sample programs, and tutorials.

Code Guidelines, Templates, and Sample Programs

Code guidelines based on previously published documents [Ell92] gather rules and recommendations as well as lists of common pitfalls and suggestions for avoiding them. Code templates exist for both supported programming languages.

Some introductory tutorials have already been developed within the collaboration, but these will be extended to cover a wider range of topics including the selected OO methodology and notation, the CASE Tool, and C++ language tutorials for more proficient programmers. Other vendor tutorials will be combined with those written within the collaboration.

Quality Assurance

Our software quality assurance (QA) strategy is based on a stable software release mechanism with responsibility for quality assurance being distributed among several software coordinators, as detailed in Section 11.4.4.

One tool in the overall QA strategy is code reviews, in which one model under consideration is that of a mentor who can informally review new code and advise the developer of improvements, rather than a formal review process with its large manpower overheads. A bug tracking system is being evaluated that should ensure that bugs are given sensible priorities and fixed in a timely manner.

A crucial component of software QA is adequate documentation, both at the level of the user and at the level of the maintainer. There is a commitment to the creation of documentation as an integral part of the development process, not as an afterthought. Word processors have been used to create documents (the Code Guidelines being an example of this) that can be viewed both in printable form (*i.e.,* PostScript documents) and also as browseable WWW documents for online use.

Compilers, Debuggers, Application Builders, and CASE Tools

It is unfortunate that the C++ language has only a draft standard and that efficient Fortran 90 compilers are only just becoming widely available. The GNU C++ compiler has been identified as providing a suitable baseline in functionality, while excluding any extensions that are not part of the draft C++ language standard. Compilers from other vendors that conform to this minimum functionality are available on all hardware platforms being considered for use within *BaBar*. The situation for Fortran 90 is that one compiler (NAG) is available on many platforms and adheres strictly to the standards, and native compilers from the major vendors are either already released or shortly will be.

As mentioned in a previous section, some CASE tools are being evaluated, though it is realized that their use will probably be limited to a core set of developers. However, the more widespread use of the methodology and notation for which the CASE tools provide support is being promoted.

Application builders exist, although their use within HEP is not widespread. Several are being evaluated, some of which are commercial and therefore raise licensing issues. A prototype application framework that will support both batch and interactive environments is under development.

11.4.3 Data Model

The use of an object-oriented methodology for the software development environment leads naturally to the description of the data model for the experiment in terms of objects. This bypasses many of the solutions that have arisen throughout the FORTRAN era such as BOS and ZEBRA and integrates access to the data naturally into the programming language.

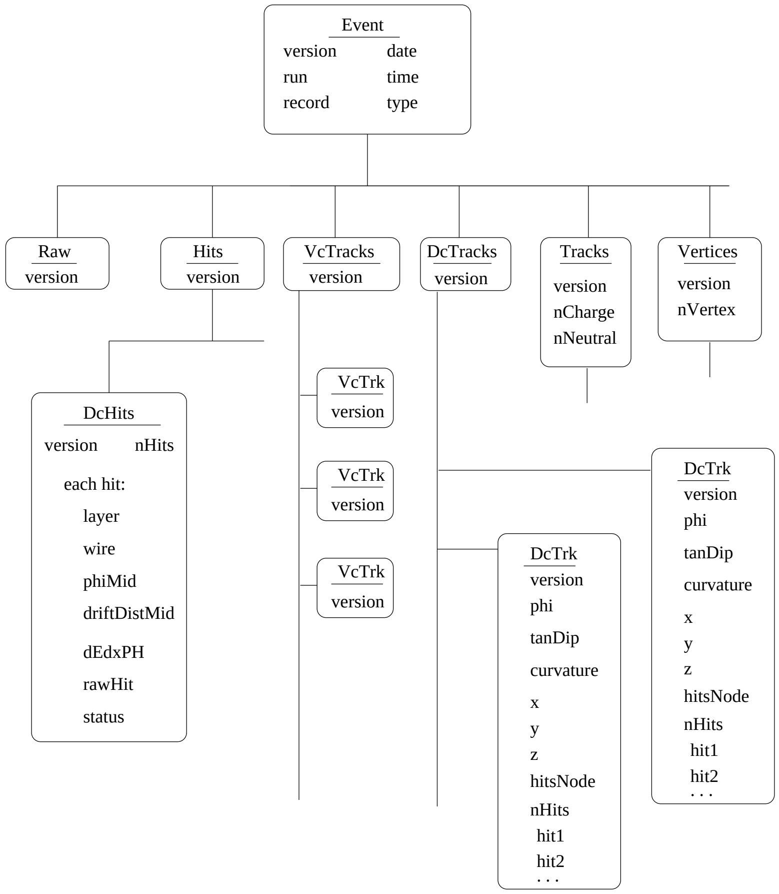

Figure 11-4. *Portion of a* FARFALLA *event tree for the* BABAR *experiment.*

A uniform data model will be applied throughout the software. Thus, the same model will apply to the access to raw, reconstructed, Monte Carlo, or calibration data, and to geometrical constants. This model will also allow for a smooth schema evolution. Advantages of this approach are that it makes possible access to the new generation of database management systems that manage objects, and that the implementation of the data storage is decoupled from its access.

An investigation of various approaches to the data model for *BABAR* has been carried out, including a number of products based on C and C++ [Sax94]. The FARFALLA [Wal94] package is the most promising existing object-oriented package on which to base the *BABAR* data model. A prototype effort using FARFALLA is underway.

The FARFALLA package provides a means to store an experiment's data on disk or tape, with convenient access in the user's program. It is based on the idea of nodes in a tree structure.

Some of the nodes in the *BaBar* prototype event tree are illustrated in Figure 11-4. As FARFALLA is written in C++, it provides a match with object-oriented design in C++ for other software. The access to the data from C++ programs is straightforward. The data itself is encapsulated (hidden), and basic access is provided via inline accessor member functions. This decouples the means of storage, which can be optimized for space, from the user interface, which can be optimized for convenience. Additional capabilities are provided by more complicated member functions. Because the detector simulation currently employs GEANT, written in Fortran, and some prototype reconstruction efforts are also being undertaken in Fortran, one of the objectives of the data model prototype is to provide access to this data. It has already been demonstrated that this can work, and current concern is in making this access in Fortran and C++ look similar.

11.4.4 Code Management and Distribution

Introduction

The code management environment must satisfy conflicting demands from the code developers, who want to be able to make changes rapidly with a minimum of controls and overheads, and the end users who wish to see a stable and bug-free code base. The situation is further complicated by the need to accommodate distributed program development throughout the collaboration and to make the resulting software rapidly available to the distributed end users. Finally, various machine architectures and dialects of UNIX must be accommodated.

Implementing a system that resolves these conflicting demands involves not only robust computer tools, but also policies on their use, and perhaps most important, acceptance by the developer and user communities that some controls need to be applied in order to ensure that the experiment produces reliable, reproducible, and defensible physics results.

Policy

The adopted policy is based on the concept of a *code package*. A package is a well-defined piece of code that has an internal coherency and well-defined interfaces to other packages. Examples of packages are track finding, vertex fitting, histogramming, *etc.* Many of these packages will come from within the *BaBar* collaboration, but others will come from outside. Individual applications are also treated as packages in this model. Each package is assigned a Package Coordinator, a responsibility that might be shared among several individuals. The coordinator is responsible for the development of stable snap-shots or *versions* of the package, ensuring that a minimum set of test procedures has been performed. Different versions for each package might be generated quite rapidly during periods of intense program development, but the development pace is dictated by the developers themselves, not outside constraints.

BaBar software *releases* are created at intervals dictated by stability and scheduling constraints. The release mechanism is controlled by a Code Coordinator. A release is formed from the most recent stable versions of every package, where stable is defined by the package coordinators. Testing focuses on interdependencies between the packages on the assumption that each package has already received some internal testing. As bugs are detected and corrected, the corrections are propagated forward to the most recent development version in a manner that does not interfere with the ongoing development. Software tools aid in the coalescing process.

Quality assurance is therefore a distributed activity, with all developers bearing some responsibility, although the main burden lies on the overall Code Coordinator and the Package Coordinators.

File Organization

A simplified file organization is shown in Figure 11-5. The directory tree is split into package and release subtrees. All software packages reside under the package tree, with the primary code repository and subtrees corresponding to individual versions. Each package has directories for source code, libraries, executables, documentation, *etc.*, for each version. Under the release subtree are stored all the complete releases, although space limitations will probably cause only the more recent releases to be maintained at any one site.

Tools

The primary code management tool is the CVS package, which allows multiple developers efficient access to a code repository while supporting tracking of code changes and their authors. The utility rCVS, developed at SLAC, extends the use of CVS to distributed developers by maintaining proxy code repositories at remote sites, via which changes are automatically propagated to and from the primary repository.

One weakness of CVS and the Unix file system is the lack of Access Control Lists (ACLs), by which individual developers may be granted access to a subset of the overall code repository. Since it is expected that most of the collaboration will act as code developers for at least one package, accidental corruption of the repository is of some concern, and ACLs could play an important role in this. Advanced distributed file systems such as AFS and DFS support ACLs and would address these concerns, while simultaneously making the code repository available throughout the collaboration without the need for rCVS. More work remains to be

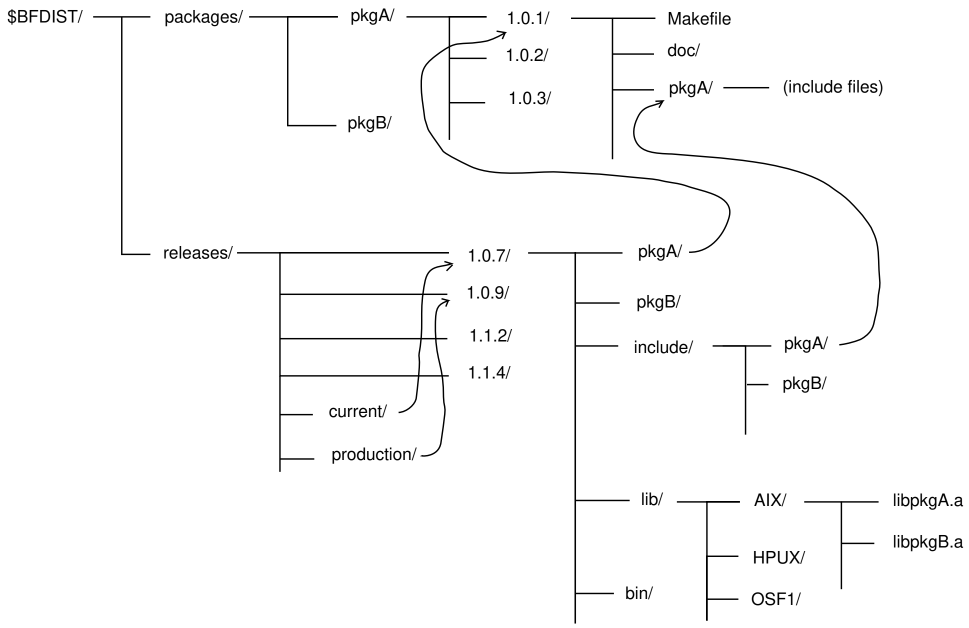

Figure 11-5. *Simplified file organization, showing the release and package subtree structure.*

done on understanding the performance issues of these file systems, and R&D is underway at SLAC with a small user community on AFS.

The various dialects of UNIX each have their own syntax for the make utility that builds the code dependencies and automates the building of application programs from the various packages or the libraries that might be part of a package. Two alternatives are being investigated in order to standardize on a version of make: `imake` (part of the X11 distribution) and GNU make.

Finally, a tool that automatically creates standardized version and release numbers and performs the functions for creation of a new release would greatly reduce the burden on the code coordinator. The Production Code Manager (PCM) prototype under development at SLAC is designed to perform these functions and is under evaluation.

Code Packages

As has been mentioned previously, a package is a well-defined body of code. Any one package is likely to be referenced by other packages. One area that needs more work is ensuring that circular dependencies do not develop between packages, since this greatly impacts the ease with which applications can be built. This implies a hierarchy in which a package can reference classes or functions from packages lower in the hierarchy, but not from its peers.

Software Release Mechanisms

A release of the *BaBar* software will consist of a self-consistent and tested set of all software packages. It will be entirely self contained, applications having been built for all supported hardware platforms, thus relieving remote sites of the burden of building applications from source code. This implies a certain minimal level of consistency in operating system versions across the collaboration with only a small set of operating system versions being supported for each hardware platform. Remote sites can either have direct access to a release using a distributed file system such as AFS, or the directory tree corresponding to each release can be made available in compact (`tar`) form suitable for shipping across a network or via tape. The goal is to have reasonably stable releases (public releases) available at fairly frequent intervals (a few weeks to months) with more stable production releases having longer lifetimes for bulk production requirements.

An area that needs further work is that of integrating the bug reporting and tracking from remote sites in such a manner that fixes or patches are made available in a timely and coherent manner across the complete collaboration.

11.4.5 Databases

The *BaBar* project faces a formidable set of information management requirements. The varieties of data that must be managed range from conventional institutional information to highly technical engineering and physics data. The computing model in which it must be managed is distributed, heterogeneous, and geographically dispersed. Modern database management systems (DBMS) will be essential to satisfy these requirements. A partial list of databases required for *BaBar* includes the following:

- Detector information: geometry constants; calibration and alignment constants; hot- and dead-channel maps; monitor logs.
- Acquisition system information: constants for front-end electronics.
- Run information: run start and stop times; trigger and readout configurations; event and record counts; tapes written; online logbook.
- Offline information: location and status of datasets; status of jobs for each run (real data and Monte Carlo).

- Tape management system.
- Document catalog: *BaBar* notes, manuals, papers, drawings.
- Engineering data: component and subsystem test and calibration records.
- Personnel information: collaboration directory, distribution lists, author list, shift list, ES&H training, human resource estimates.
- Management information: budgets, schedules, equipment inventories, software license tracking.

The possibility of storing information about interesting physics events in a database will also be investigated.

Two different database technologies are relevant to *BaBar* requirements. Relational database management systems (RDBMS) are the current choice for managing institutional data in most large organizations, including other current HEP experiments. This is a mature technology with several major vendors and many high-quality third-party products on the market, a well-established theoretical foundation, and an expressive and standardized query language (SQL) to provide portability. On the other hand, RDBMS have not been integrated smoothly into programming languages, and their simple data model (rows and columns) is poorly matched to the complex data structures needed in a HEP environment and supported by an object-oriented language such as C++.

Object-oriented database management systems (OODBMS) provide an increasingly popular alternative to RDBMS. These systems generally map database elements directly to software objects. Most of the complexities of dealing with the database management system are hidden from the programmer. Unfortunately, these products are less mature and lack both a firm theoretical foundation and (as a consequence) satisfactory *ad hoc* query tools. Moreover, though several candidate interoperability standards exist, there is not yet strong, cross-vendor support within the industry for any of these specifications.

Oracle is already in place at SLAC and is in use by both *BaBar* and PEP-II for a variety of different types of institutional and engineering data. Most such data will likely remain in Oracle at least until OODBMS with equally expressive query languages become available. *BaBar* will investigate the use of OODBMS for those databases accessed heavily from physics programs, *e.g.,* calibration and alignment constants, channel maps, *etc.*

Some types of data, such as the run database, are likely to be frequently accessed both from within the physics code and from external query tools. The choice of technology for such databases will be deferred until more experience with OODBMS is gained and a better idea of the access requirements of the physics code is obtained. It is likely that the physics code will require access to at least some relational databases, however, so *BaBar* will also select, or if necessary, develop a class library for accessing a generic RDBMS.

Large scale databases represent potentially major investments, both for licensing and administration. It is, therefore, unlikely that all collaborators will have local access to all database software. Various models can be imagined whereby data is shared with other collaborating institutions. For cases in which database software must be present, site licensing will be investigated, and a mechanism for periodically replicating and distributing certain databases will be developed. Where licensing is not essential or is impractical due to expense, services will be provided to access data over the network either via *BaBar*-developed client tools or via low-cost, third-party "data-dipper" applications.

11.4.6 Graphics

Computer graphics serve an important role in various stages of an experiment. During detector design, they aid the physics simulation tasks and CAD. In the course of code development, graphical detector representations and interactive event displays are necessary for pattern recognition and event reconstruction studies. For physics analysis of data, it is essential to have tools for event viewing, parameter representation, multidimensional statistical analysis, and histogramming. Graphics displays with graphic editors are useful for producing high-quality hard copies for presentation and publication. Also, graphical user interfaces (GUIs) are important for human interfaces in a variety of contexts, such as basic computer interaction (the desktop), software engineering tools, job submission, and detector control and monitoring. In this section, the available computer graphics and GUI options are summarized.

Graphics Hardware

Compared to the industry standards, the hardware requirements for graphics in high-energy experiments are moderate. The major factors are compute performance, variety of color display, screen resolution, and the ability to produce quality hard copies. For screen display, there exists a range of workstations with sufficient performance for most applications. Their price/performance ratio is improving with time. For hard copies, color printers and color copiers are becoming inexpensive. These trends are expected to continue.

Software Options

For histogramming and plotting curves, there are several packages already available such as Handypak and Topdraw, along with the CERN graphics packages HBOOK, HPLOT, PAW++, *etc.* For event and detector displays, an assortment of approaches ranging from simple two-dimensional X11 or PostScript cartoons to complex three-dimensional structures can be useful. Most three-dimensional packages have high manpower overhead, and options should be examined carefully. One could also adopt a package like that of ALEPH's V-plot for representing three-dimensional data of events [Dre91].

In the UNIX environment, there are a number of commercially supported three-dimensional graphics packages available such as Graphics Kernel System (GKS), Programmer's Hierarchical Interactive Graphics System (PHIGS and PHIGS PLUS), Silicon Graphics Library (Open GL), and Renderman. While X11 and PHIGS can be obtained free of charge from the Open Software Foundation, other packages must be purchased from commercial vendors. PEX (PHIGS Extensions to X) also supports three-dimensional graphics within the X-Window environment. Some of these packages have ISO and/or ANSI standards defined. They also provide device-independent interfaces. In the near future, a choice among these packages will be made based on relative performance, ease of use, and portability.

There are also a number of visualization packages available on the market such as AVS, apE, and PV. With these packages, the user can develop graphical displays interactively using the mouse and keyboard at a workstation.

Graphical User Interfaces

A graphical user interface (GUI) is characterized by pointing, clicking, and dragging with a mouse; pull-down menus; dynamically resizable windows; and other features. GUIs offer the user a high degree of interaction and assistance in navigating through unfamiliar programs. Each component of the program can be associated with graphical icons illustrating the intuitive connection with its associated function. A properly designed GUI can dramatically aid the training to operate a program, reduce errors, and increase operator efficiency.

There can be a considerable cost associated with these user interfaces in terms of programming development. In fact, there are reports that 80–90% of a program's development may be required to design and produce a good GUI in the absence of a good interface builder. The availability of commercial packages called interface builders significantly eases the work load in developing GUIs. Products producing X11-based GUIs are numerous, including XVT-Power++, TAE+, UIM/X, Garnet, and TeleUSE.

11.5 Online System

11.5.1 Introduction

The online system provides for the operation of the *BaBar* detector while it is preparing for or engaged in data taking. The system's primary responsibility is for collecting and storing the experiment data. It will also monitor and maintain the integrity and quality of that data. The online system must maintain a high level of availability during collider operation. This system can be organized into several main areas as shown in Figure 11-6 and listed below:

Experiment Control. Controls and monitors the environment of the detector.

Run Control. Organizes, controls, and monitors the detector components, hardware, and software for data collection.

Data Flow Control. Controls and monitors (in real-time) the passage of data from the detector subsystem back-end electronics to the final data repository and other data consumers.

Trigger Control. Controls and monitors the decision making processes that determine whether data will be recorded.

Clock and Control. Provides a variety of precise timing signals and strobes to the subsystems.

The online system must interact with and link together the various components of the *BaBar* detector as illustrated in Figure 11-7. Some of the systems illustrated, such as the PEP-II control system, require only a communication protocol specification from the online system since they are external to the detector system's responsibilities. Other subsystems, such as the detector front-end electronics, require detailed specification so a complete range of operations is allowed. The online system will share much of its architecture and software support with the offline system.

The online system will use an object-oriented design methodology for many of its components. This approach stresses the reuse of computer code, logical constructs, and ideas.

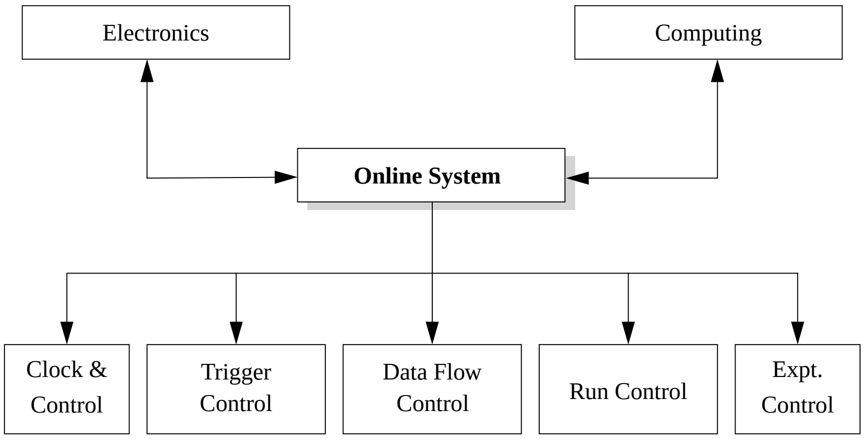

Figure 11-6. *Schematic organization of the online system.*

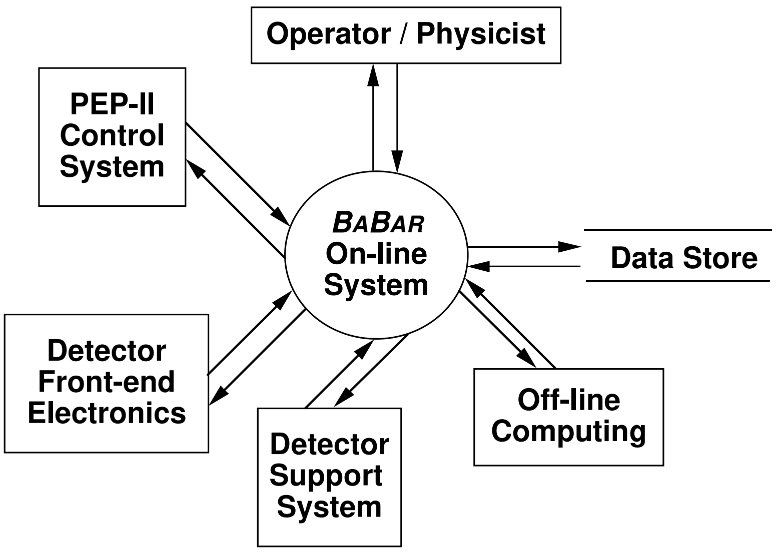

Figure 11-7. *Context diagram of the online system.*

Within this methodology, the system will develop iteratively with each major release usable as a test system, but without complete functionality. Later versions will build on the existing system by filling in missing functionality. This model furnishes concrete deliverables for test stands, early beam tests, the high-energy ring commissioning, and the colliding beam commissioning. It also provides for adaptive improvement of and evolution in the system.

The description of the online system that follows presents different views of the system. The logical view focuses on the functional description of the system. The subsystem view details the interface to the detector subsystems and their common logical and physical constructs. The operator view describes the system's appearance to and allowed interactions with the physicist/operator. The implementation view describes the working decisions and specific implementations of the system. This approach aids in coping with some of the system's intrinsic complexity by focusing on only one aspect of it at a time.

11.5.2 Logical View of the Online System

The online system will use a client-server, distributed-object model for its architecture in which high-level control of the system is arbitrated by a series of console, manager, and interface processes (Figure 11-8). A console process such as a GUI program or a script always acts in the client role and allows the user to interact with and exert direct external control over the online system. A manager process provides resource allocation and control for the system's consoles, device interfaces, and other managers, and is composed of objects using Common Object Request Broker Architecture (CORBA) [OMG91] techniques to implement client-server communications. A device interface layer with well-defined functionality controls the system's low-level distributed devices, encapsulates the associated electronics, databases, and processes, and shields the rest of the online system from these complexities. In this architecture, multiple clients may need the services of a specific manager, but only one client at a time is allowed control. Token passing controlled by specialized managers called brokers will be used in this role.

Experiment Control. Experimental Control processes ensure correct functioning and efficient performance of the BaBar detector by continually monitoring its components. This involves periodic readback and analysis of the detector electronics and diagnostic procedures on the detector components. Experiment Control is not responsible for the data flow, trigger, or clock and control areas, which have their own control processes. Since the functionality of this system is common to most other high-energy physics experiments, the EPICS tool kit will form the basis of the Experiment Control framework. EPICS is freely available software developed originally at LANL and later at ANL and CEBAF. It was originally designed as an accelerator control system, although its current use includes several *experiment control*

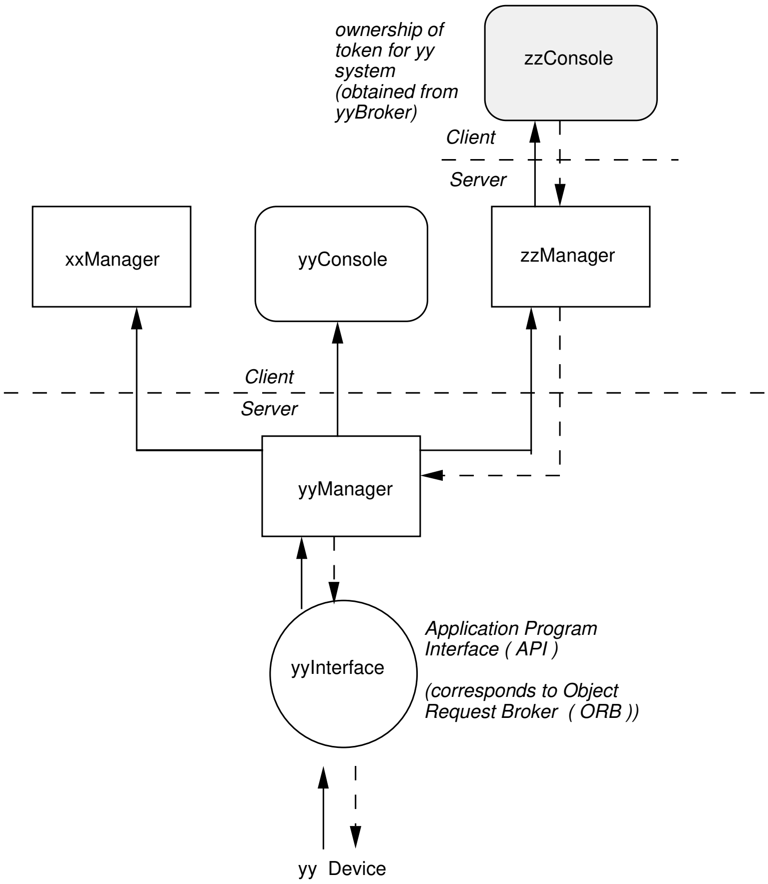

Figure 11-8. *Interaction of multiple consoles/managers, logical layers, and interfaces.*

systems. Because EPICS is intrinsically a procedurally designed and implemented tool kit, a manager and interface software layer will be built above it.

Run Control. Run Control will guide the operator along the proper operational sequence by which the detector is readied for data collection, calibration procedures, or diagnostic procedures. Crucial hardware and software components will be automatically tested. During data taking, Run Control will continually monitor the detector and experiment data to detect failures and malfunctions. If an error is detected, an expert system will be used to help diagnose and fix the problem. This system is expected to evolve so that it can solve some routine problems on its own without involving the operator.

Run Control organizes and controls the detector for data taking using a configuration manager and a partition manager. It also has an expert system manager which aids the operator in diagnosing and fixing problems. Configuration management provides the mapping among the physical hardware/software components and the appropriate control and monitor managers. All persistent attributes of these components will be stored in centralized object-oriented databases. For example, the device interface will build its EPICS databases from the central database according to the hardware needs. In particular, each component will be described by a state transition diagram. Partition management provides dynamic reorganization of the system's components into independently operating logical subsets. For example, the main partition could include the trigger and other system-wide resources, while several subsystem-level and individual readout crate partitions also exist. Configuration of a particular partition requires the instantiation of the objects representing the various hardware and software components on a workstation and a selective loading of data files and code to the associated embedded CPUs.

Trigger, Data Flow and Clock and Control. Trigger, data flow and clock and control are described in the Chapter 10. From the perspective of the online system, the trigger controls the data flow in the experiment. Trigger signals in the form of accepts and rejects, busy and ready, success and failure, are distributed as strobes via Clock and Control.

11.5.3 Subsystem View

Subsystems have several interfaces with the online system. Front-end electronics provide input to the trigger and data flow processes as described in Chapter 10. General purpose hardware and software perform routine monitoring and control functions. A set of special purpose hardware and software performs diagnostic testing for each subsystem. All devices are associated with a standard readback identification and status either directly or through the controlling processor. The identification is used for inventory, maintenance of repair records, and configuration details.

Most of the controlled and monitored components, including some purely logical components, can be described by a state transition diagram which has a direct EPICS implementation. The status is a state-dependent qualifier that describes the device's readback in terms of the device's expected behavior in that state.

The device manager drives the device interface through its transitions, enforcing an internal logical consistency and conformity with external constraints. For example, the magnet power supply in the state OFF is expected to have a current output of zero, within the tolerances of the readback. If the current monitored is not zero, then the returned status would be OVER CURRENT and color-coded red to signify a possible malfunction. The set of status values will depend on the allowed states of the device.

Data retrieved from the monitoring processes will be archived and error and message logs maintained. Status and state changes will be monitored by Run Control to maintain the integrity of the experiment data. During normal data taking, special readout options will be available. Nonsparsified data from a subsystem can be collected and monitored. This might check the validity of deleting below-threshold data from the normal data stream. A snapshot mode can monitor data from the VRAMs. This mode might check feature extraction algorithms by allowing comparison with full wave forms. An analysis mode can save all data buffers associated with a given trigger, so that the history of an event in the data acquisition can be fully reconstructed.

The process of controlling a device involves changing its state or the values of its functional parameters. A manager acquires a control token for the device and then takes the desired action. During the transition, a graphical representation of the action and its progress will be continually updated and displayed. For complex collection objects, the manager will enforce any necessary synchronization and present the most relevant progress display. Any error messages generated are sent to the operator and Run Control and are archived.

Most detector components will have associated calibration and diagnostic procedures to ensure their correct performance during operation. As required by partitioning, all subsystems must be capable of independent and concurrent calibration. In addition, individual readout controllers must be allowed to run in diagnostic mode while the others are participating in normal data-taking operations or calibration. As a result, a controller or set of data-flow strobe signals may either be replicated, or be made part of a diagnostics partition to allow independent operation.

A parallel load test will also be provided, using the controller to issue data pulses and triggers, which exercises the entire normal data-taking stream of the main partition.

11.5.4 Operator View

The online system presents an operator of the detector with a standardized, hierarchically arranged set of control, diagnostic, and monitoring panels in which the most important parameters and controllers are also those most prominently displayed. Control panels allow an operator to perform and monitor normal operations on a particular subsystem. Diagnostic panels allow an operator to perform normal operations as well as the capability to test and exercise individual subsystems. Monitor panels only display information about a subsystem and do not allow an operator to perform any control functions. Detailed information and control of each element in a subsystem will also be available, but only when requested explicitly by the operator. The system will rely on the standard graphical user interface libraries and tools of the X-Window/Motif system.

For example, the calorimeter subsystem might have a main control panel that includes the status of the bias voltage power supplies with an overall gain knob, the most recently performed diagnostics, a summary of the average hit rates in the crystals, the status of the calorimeter's front-end electronics and DAQ modules and a set of histograms reflecting the overall performance and recent history of the calorimeter's functioning. The bias voltage power supply status would, for example, allow access to each of the individual power supply crates, which would in turn allow access to individual bias voltage channels through a standard point-and-click interface.

11.5.5 Implementation View

The ingredients for implementation of a successful online system must not only meet the technical and operational requirements, but must, in addition, contribute to a coherent overall architecture. Options available today are considerably different from those available only a few years ago. Rapid advances in electronics and computer technologies are likely to continue changing the landscape over the design and implementation period for the *BaBar* detector systems. Thus, these ingredients must exhibit sufficiently general properties so that they and other ingredients may be changed relatively freely and without requiring large architectural changes to the system. To this end, a preliminary selection of online components has been identified.

Industry standard VME will provide the backbone of the online system. VME is a mature technology in terms of reliability and availability of products. Its backplane performance is satisfactory for control applications and many data applications. Enhancements to a VME system are possible via VME-64 and/or VXI. Various auxiliary buses (*e.g.,* CAMAC) and cable interconnects (*e.g.,* MXI) can be used in conjunction with VME.

Single board computers residing in VME crates will assume the role of intelligent crate controllers. The need to provide portable online code implies the use of coding standards to avoid system-specific extensions, and the use of a real-time kernel to prevent code from being dependent upon the hardware specifics of a particular single board computer. The VxWorks kernel satisfies these requirements with excellent real-time performance, ANSI C/C++ language support, and compliance with the POSIX standards.

A framework within which to build an online system, perform overall control functions, and collect and analyze monitor data will be based on the EPICS package. EPICS is based upon an architecture of UNIX workstations and single-board computers running VxWorks connected by a network running the TCP/IP protocol. In its basic form, EPICS includes a communication system, a database, periodic monitors, a sequencer, and various device drivers for specific VME data acquisition cards. The toolkit also includes a number of applications for handling alarms, archiving data, editing the database, building new applications, *etc.* Various third party software packages (*e.g.*, Mathematica and MatLab) interface to EPICS.

Another aspect of software development concerns the quality of design and maintainability. Consistent with overall computing guidelines, the online system will be based upon the object-oriented philosophy, utilizing the C and C++ programming languages. The ANSI C language will be used for compatibility with the EPICS package, while C++ will be used for higher level software components. CASE tools will be employed where applicable to assure sound software design.

Various networking technologies are candidates for the online system. The TCP/IP protocol is necessary for compatibility with the EPICS system and the rest of the SLAC site. Currently, both Ethernet and FDDI offer acceptable solutions for control and slow monitoring applications. It is expected that emerging technologies such as ATM (Asynchronous Transfer Mode), FCS (Fiber Channel Standard), and SCI (Scalable Coherent Interface) may become realistic candidates in the timescale of *BaBar*; such a solution might address both data acquisition and slow control needs. Bandwidth and latency are the primary issues facing the network system; these are discussed further in Chapter 10.

Generally, solutions for the online system will be chosen so as to be compatible with or identical to those for the offline computing system whenever reasonable. Mass storage, for example, will need to be equally available to both; thus a common network solution is desirable. Real-time event reconstruction will require executing the full offline code in the online environment, thus a standard workstation environment is needed.

11.6 Reconstruction and Analysis Framework

11.6.1 Scope

A common framework for application programs has to encompass several areas of deployment. For the data acquisition, trigger, and online systems, it must provide support for the monitoring processes and for the trigger algorithms within the online trigger farm. Within the offline environment, it must provide support for Monte Carlo simulations, the bulk processing and reduction of data, and physics analysis. This implies that the framework must provide support both for input of data from a variety of sources including disk and tape files and the online event server and for output to a similar set of permanent and transient destinations. Furthermore, it must provide an interface that is suitable for use in both interactive and batch environments, and ideally support both expert and novice users.

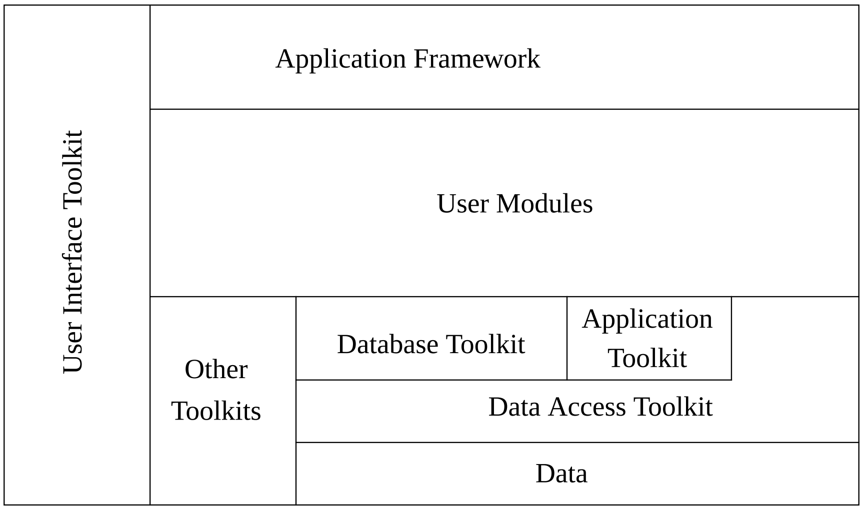

Figure 11-9. *Software framework.*

The question arises whether a framework is just a class library (using object-oriented terminology) or whether it encompasses greater functionality. The key distinction between a framework and an arbitrary collection of classes, however closely related those classes might be functionally, is that a framework describes not only the objects, but also their interactions with each other in a domain-specific manner. An application may therefore consist of several different frameworks, one providing access to the event data, another providing histogramming support, all under the overall control of the application framework. A framework that is available purely as a class library is called a toolkit.

11.6.2 Conceptual Overview

Our concept of an application framework is illustrated in Figure 11-9. User-written code, in the form of modules, can be incorporated with other modules under the overall control of the application framework, but with access to lower level frameworks and toolkits. Modules can be associated into paths, multiple paths being supported, where each path could correspond to a particular physics process, for example. The framework allows the user to select an input module and will then execute the modules in the various paths in the appropriate sequence. Filter modules may terminate or redirect processing of events in a particular path if the data do not meet the specified selection criteria. Events that reach the end of each path may be directed to a set of output streams, allowing for simultaneous stripping of selected data samples in the same pass through the data.

11.6.3 Toolkits

Several toolkits have been identified, including the application, data access, database, user interface, and visualization toolkits. The application toolkit is part of the framework itself and provides mechanisms whereby user code can find out attributes of the environment such as the run number and the presence of other toolkits. The data access toolkit provides access to data within an event from both C++ and Fortran, and the database toolkit provides access to the calibration, geometry, and other databases that specify the detector environment. The user interface toolkit allows users to prompt for operator input and to display quantities of interest, and the visualization toolkit provides a method whereby event data may be displayed graphically in a variety of modes. Thus, a track may be displayed in a Cartesian coordinate system, or as residuals from the fitted space points.

11.6.4 Framework

A framework that addresses the requirements is under development. It is written in C++ and uses the Tool Command Language (Tcl) [Ous94] as its primary user interface. It thus provides a flexible text-based interface that supports scripts and is therefore suitable for the expert user and for a batch mode of operation. It is intended that a graphical front-end to this framework based on the Tk package be created. Expect [Lib91] is a suitable candidate for this. The main focus of the development for this framework prototype is the data model prototype discussed elsewhere. In parallel with this development, several third-party application frameworks are being evaluated, including both public domain and commercial products. One problem appears to be their use in a batch environment, but they offer sophisticated user interfaces when used interactively.

11.6.5 Application Builder

The overall concept of the framework is that an application can be built that contains a large set of modules, only a few of which may be selected if so desired at run time. Thus, the same application may be used for reconstruction, diagnostics, or analysis depending on which modules are enabled. This model depends on there being dictionaries of standard modules that may be combined with user-written modules. It is possible that shared libraries may also be used to this end. Some of the application frameworks under investigation support the graphical creation of applications, and this capability has been shown to greatly improve the user friendliness of such systems.

11.6.6 Standard Applications

Several standard applications will be developed. One of these will be the stand-alone event display, although it is expected that an event display module for incorporation into other applications will also be available. Another application will support the dumping of event data in a variety of formats as a diagnostic aid. The applications running in the online trigger farm and for the main offline data processing may themselves be considered to be standard applications, and therefore available, in order that tests of the trigger efficiency and other systematic studies may be performed.

11.6.7 Job Submission

Much of the more time-consuming data processing will take place in a batch environment, and the application framework should interact seamlessly with the supported batch systems. Basing the framework prototype on Tcl was motivated by its availability as a scripting language within the UNIX shell environment as well as its use as a command parsing toolkit within applications. The generation of job submission scripts, spanning multiple jobs as well as single job submission, should therefore be a well-supported operation, capable of good integration into the overall batch environment.

11.6.8 Bulk Production

The submission of jobs for the bulk reduction of the raw data and generation of DSTs requires a level of support much greater than for casual job submission. Databases will need to track which data have reached which stages of the reduction, and which jobs failed and need further investigation. It might be possible to provide some tools to aid in quality checks over and above scanning of many histograms. Rapid feedback and high quality assurance are crucial to ensure that problems are detected quickly and do not affect the overall physics results.

11.7 Computing Support

11.7.1 Introduction

Support of the *BaBar* computing environment will be split essentially two ways. The collaboration will provide support for experiment-specific tools and the online system, as well as for offline analysis software. A substantial part of this effort will be handled using computing resources at remote institutions.

Infrastructure support including computing hardware, networking, and system software will be provided by the local computing center and/or departmental staff. SLAC Computing Services, for example, provides this support at the SLAC site.

11.7.2 Collaboration Support

The collaboration must design, maintain, and document the online and offline software framework, data model, analysis environment, and user interfaces to the various tasks. The personnel model is similar to recent typical approaches in HEP, in which physicists write most of the analysis and reconstruction code, including many of the peripheral tools and scripts, and software engineers address the other aspects where necessary, using appropriate commercial design tools. Examples include database technology and the design of the software framework. This model has the benefit that the physicists will have close contact with the physics aspects of the code, while the computer science and systems aspects are handled by people with the requisite expertise.

It must be stressed that the software for the *BaBar* experiment represents a very large system, and there are several major tasks that require special attention. The support for these activities is expected to consume a significant amount of manpower within the collaboration. Most of this manpower will consist of physicists and students, and each collaborating institution is expected to contribute a significant portion of its manpower to software development. The use of software engineers will have to be limited to those tasks for which they are essential, in order to minimize costs. Five full-time-equivalent (FTE) engineers are thought to be required. The following is a list of some of the major support tasks, most of which include some amount of infrastructure development, database technology, and coordination.

Project Computing Engineer. One of the FTEs mentioned is at a senior level to serve as the Project Engineer, responsible for the entire computing system.

UNIX System Management. There will be a UNIX system manager, responsible for the smooth running of the computing hardware, particularly online.

Code Management. Code coordinators will be appointed for all software packages, such as the online system, event reconstruction, and Monte Carlo. Their responsibilities include supervision, quality control, and management of production releases.

Data Processing. In the course of the experiment, a very large amount of data will be collected. The production of DSTs is therefore a major undertaking that needs careful management by data aides under the supervision of a physicist.

Data Distribution. Large amounts of data in DST form will need to be distributed to and received from various remote sites as discussed earlier. Data aides monitored by a physicist will perform this task.

Database Management. The *BABAR* experiment will have a large number and variety of databases. Database management is not a trivial task, and this will require at least one dedicated expert.

Documentation and Training. Documentation methods will be developed and enforced. Various topics may require training, such as C++ and databases. Identification of these topics and organizing training sessions is necessary.

Tools and Utilities. The experiment will require a great variety of tools and utilities. Some will be designed and written. All must be subsequently maintained.

11.7.3 Infrastructure Support

Infrastructure activities are expected to take place within all collaborating institutions, and these activities must be coordinated between institutions and with the collaboration. This coordination will be facilitated through a regional representative committee within the collaboration and through regular meetings between the collaboration and local support organizations. An example of the latter is already in place at SLAC where the majority of computing resources will reside. It is expected that similar arrangements will also be put into place at other regional centers.

An active discussion with SCS has been underway to identify the support that they will provide to the BaBar collaboration. A detailed proposal defining possible infrastructure support is given in Reference [SCS94] and is summarized below.

SCS will support the computing hardware, including CPUs, networking, and storage. This includes much of the offline environment and the network components at the detector. It also includes certain aspects of the Level 3 farm itself, which is envisioned to consist of general-purpose workstations. Hardware support includes planning, purchasing, installation, operation, tuning, and maintenance. Software aspects supported by SCS include system management of the Unix workstations and the X/Motif graphical environment. In addition, a number of general programming and database tools will be managed by SCS, including compilers, editors, graphics, and Oracle. Software support also includes management of user accounts, disk and other resources, backups, a batch system, *etc.*

11.8 Integration Issues

The online computing system will be an important tool in the commissioning of the BaBar detector, and even of the PEP-II accelerator. The high-energy ring is scheduled to begin commissioning approximately a year prior to the completion of the low-energy ring. Background conditions will need to be studied during this period. The interaction region commissioning will involve simple detectors in advance of BaBar which will require some data acquisition. The essential components must, therefore, be ready relatively early, and this also has implications for the schedule for the control room and electronics areas.

The Level 3 data acquisition farm will include peripherals and CPUs, as well as a data link to the SLAC computer center. Operator interfaces will be discussed below. There already exists a fiber optic cable between the IR hall and the computer center that meets the requirements. The CPUs and peripherals should be installed in a clean, air conditioned environment with only occasional access necessary. The space for approximately four 19-inch racks is required—two for the CPU farm itself, and two for the peripherals.

Another crucial aspect for the effective running of an experiment is placement and number of computers and displays for control, monitoring, and debugging. Approximately ten monitors will be required in the control room to permit simultaneous development and control activities, and to dedicate screens to displays from the accelerator, environmental monitoring, event display, and run status. Several additional displays will be required in the electronics area to permit effective debugging and maintenance of the apparatus.

An external system that must be linked to experiment control is the PEP-II accelerator. The PEP-II Control System will be an extension of the SLC Control System. However, new developments such as longitudinal feedback and rf control will be based on EPICS, with the IOCs (Input/Output Controllers) integrated and supported by PEP-II Controls. At this time, the role of the PEP-II Controls VAX is seen as limited to that of channel access client; data from the IOCs of the *BaBar* detector are thus directly available to PEP-II Controls. Conversely, information stored in the VAX database would be sent via the network to the *BaBar* Online System on a periodic basis (on a likely timescale on the order of minutes). This information would include beam position monitor data, beam energy, lifetime, current and emittance data, local pressure gauge readings from the ring vacuum system, *etc.* This data will be entered into the *BaBar* PEP-II database and be available for history plots and other display options.

Circumstances that may require a more immediate response than is possible with the network link to an EPICS controller are envisioned. One requirement is to assist in the process of beam tuning, both to reduce backgrounds and to increase luminosity. In response to a request, the online system will provide fast digital signals from detector chambers for noise metering, as well as signals from the detector luminosity monitor.

For proper control of both the detector and the accelerator, the state and status of many components must be monitored and the information shared between the online and the PEP-II Control Systems. Control over energizing the *BaBar* solenoid, over allowing access to the interaction region, and over allowing injection of beam into the ring, will be allocated on the basis of permissives exchanged between *BaBar* and PEP-II Controls. Information about the presence of colliding beams, the active logging of data, the value of the solenoidal field, *etc.*, will be made available to both control rooms. During injection, radiation damage to detector components will be minimized by reducing the injection rate if a threshold is exceeded in a PEP-II ionization detector at the IR.

11.9 System Responsibilities and Management

The overall responsibility for the *BaBar* computing system rests with two system co-managers and the computing project engineer. There will be designated people responsible for the various elements of the system, including, for example, code coordinators for the various packages, an online coordinator, *etc.*

Broad policy, communication, and planning issues are addressed by three committees:

- The detector subsystem (vertex chamber, drift chamber, *etc.*) reconstruction, simulation, calibration, and diagnostic code will largely be written by people closely involved in the design and construction of the subsystem. A committee of representatives from each of the subsystems will be responsible for ensuring a uniform approach within the context of the computing system and to allow communication of requirements.

- To ensure close and effective integration of computing efforts throughout the collaboration, a Regional Computer Representative Committee has been formed. This group brings together the Computing System Managers and Engineers plus a representative of each socio-geographical region in the collaboration. The group provides a forum for raising and solving problems of particular concern to collaborating institutions. It also provides a channel of communication from the Computing System to the widely spread computing effort of the collaboration.

- Finally, a planning committee exists with membership from both SCS and *BaBar*. This group is considering the support issues and resource requirements and is an important input to SCS for their planning and budget requests, and for coordinating which requirements will be the responsibility of the collaboration and which of the computing center.

References

[Aba94] F. Abar and W. Lidinsky, "The Effects of X-Window HEP Graphics Applications on ESnet," presented at CHEP94, San Francisco, California (1994).

[Ast92] D. Aston, A. Boyarski, A. Breakstone, D. Cords, T. Glanzman, F. Harris, T. Hung, P. Kunz, and F. Porter, "Computing for a B Factory," *BaBar Note* # 82, (1992).

[BaB94] The *BaBar* Collaboration, "Letter of Intent for the Study of CP Violation and Heavy Flavor Physics at PEP-II," SLAC–443 (1994).

[Bad90] A. Baden and R. Grossman, "A Model for Computing at the SSC," Superconducting Super Collider Laboratory Technical Report No. 288 (1990)
C. T. Day *et al.,* "Database Computing in HEP—Progress Report," in Proceedings of Computing in High-Energy Physics (1992)
C. T. Day *et al.,* "The PASS Project Architectural Model," in Proceedings of Computing in High-Energy Physics (1994).

[Boe94] C. Boeheim, K. Gounder, and R. Melen, "Computing Model Options for B Factory," *BaBar Note* # 154 (1994).

[Bos94] Talk presented by Rebecca Bostwick at the ESnet Site Coordinating Committee at CEBAF (1994).

[Boy90] A. Boyarski, T. Glanzman, F. A. Harris, and F. C. Porter, "Report of the Computing Group," in Proceedings of the Workshop on Physics and Detector Issues for a High-Luminosity Asymmetric B Factory at SLAC, ed. by D. Hitlin, SLAC–373, LBL–30097, CALT–68–1697, 575 (1990).

[Dou94] D. Doughty Jr., D. Game, L. Mitchell, G. Heyes, and W.A. Watson III, "Event Building Using an ATM Switching Network in the CLAS Detector at CEBAF," presented at the International Data Acquisition Conference on Event Building and Event Data Readout, Batavia, Illinois (1994).

[Che93] H. Chen, J. Hutchins, and J. Brandt, "Evaluation of DEC's Gigaswitch for Distributed Parallel Computing," SAND93–8013B, Sandia National Laboratories, California, (1993).

[CLE94] Information concerning CLEO-II computing was kindly provided by M. Ogg, J. Urheim, and A. Weinstein.

[Dre91] H. Drevermann *et al.,* in "Computing in High-Energy Physics '91," ed. by Y. Watase and F. Abe, Universal Academy Press, Tokyo, Japan (1991).

[Ell92] Ellemtel Telecommunication Systems Laboratories, "Programming in C++: Rules and Recommendations," Sweden (1992).

[EPI92] "Experimental Physics and Industrial Control System (EPICS)," Argonne National Laboratory and Los Alamos National Laboratory (1992).

[GIG93] Digital Equipment Corporation, "GIGAswitch Systems Manager's Guide," Part No. EK–GGMGA–MG.A01 (1993).

[Gou94] K. Gounder, S. Bracker, K. Hendrix, and D. Summers, "A Simple Multiprocessor Management System for Event-Parallel Computing," *BaBar Note* # 128, (1994).

[HUB94] Alantec Corporation, "PowerHub Reference Manual," vol. 2.2, Rev. C, Issue 1, Part No. 400–0336–000 (1994).

[Lib91] "Expect: Scripts for Controlling Interactive Programs," Don Libes, Computing Systems, Vol. 4, No. 2, University of California Press Journals (1991).

[NCD89] "Network Loading and the NCD Network Display Station," Note 3, available from NCD, (1989).
F. Abar, C. O'Reilly, and E. Wicklund, "Network Load of X Terminals at CDF," FNAL Report, (1992).
R. Cottrell, "X Terminal White Paper," SLAC (1993).

[OMG91] Object Management Group, The Common Object Request Broker: Architecture and Specification, Revision 1.1, OMG TC Document 91.12.1 (1991).

[OOM94] G. Booch, "Object Oriented Analysis and Design with Applications," 2nd ed. (Benjamin Cummings, 1994) ISBN 0-8053-5340-2.
D. Coleman, "Object Oriented Development, The Fusion Method" (1994)
P. Desfray, "Object Oriented Modeling," (Addison Wesley 1994).

[Ous94] J.K. Ousterhout, "Tcl and the Tk Toolkit" (Addison-Wesley, 1994).

[Por93] F. Porter, "*BaBar* Event Size Estimate," *BaBar Note* # 108 (1993).

[Sax94] S. Saxena and T. Glanzman, *BaBar Note* (in preparation).

[SCS94] "SCS Support of the B Factory," *BaBar Note* # 145 (1994).

[Wal94] C. Walter and B. Nolty, "The FARFALLA User Guide, FARFALLA; V. 1.5" (1994).