Instructional computer systems for higher education

by CHARLES J. PRENNER
University of California, Berkeley
Berkeley, California

and

ALFRED Z. SPECTOR
Harvard University
Cambridge, Massachusetts

ABSTRACT

In most universities, increased instructional utilization of computers by many departments has been the rule. With the associated diversity of instructional requirements and evolving hardware capabilities, the choice of a proper computer system for this environment has been made more difficult. In this paper, a review of the requirements for such a system is presented and the alternatives are analyzed in light of these requirements. It is shown that non-interactive systems are the least desirable educationally and furthermore, that the economic justification for their use is no longer as strong as in the past. In support of this, a description of an inexpensive interactive system currently in use at Harvard is given.

INTRODUCTION

With an increased need for instructional computing in higher education and tighter university budgets, educational administrators must determine which computer systems provide the highest educational benefits at the least cost. Questions of interactiveness, size and performance must be considered. They can only be answered by an analysis of the user community, of the various possible alternative systems, and of the costs of these systems in light of the conditions on a particular campus.

THE USER GROUPS

In order to determine the desirable attributes for an instructional system, a survey must be made of the various user groups and of their individual requirements. Unlike many systems where essentially one class of end-user is supported, educational systems typically must host a widely variegated user community.* It is convenient to classify the users into four categories: (a) those who utilize the computer for introductory programming courses, (b) users from intermediate and advanced computer science courses, (c) users from non-computer science courses utilizing the computer as an instructional aid or tool, and (d) users consisting of instructors and staff. Each group is analyzed in turn.

Users from introductory computer science courses

This group utilizes the computer to obtain a general understanding of the techniques of computer science. In this category are general interest students, future computer science majors and students from the physical or quantitative social sciences. Although it is true that the individual goals of the students may vary, most students share the same computing requirements. Mainly, they must all cover a large amount of information in a short time. They must also be given a broad and balanced view of the subject matter. Finally, because of their lack of experience, they require especially simple system conventions and a protected environment which provides for easy detection and correction of errors.

A computer system that suits their needs must be capable of supporting a suitably rich set of programming languages as well as providing reasonable conventions for their use. Since introductory students will undoubtedly make many mistakes, easy debugging of programs and clear diagnostics must be provided. Finally, as all neophytes have difficulty in coping with what appears to them to be convoluted conventions, these must be held to a minimum.

* Computer aided instruction, (CAI) *per se* is considered to be a different use of computers and is not considered in this paper.

Users from non-introductory computer science courses

This group requires a wide variety of languages in which to program and the ability to develop, debug and test programs as quickly as possible.

For example, students in this category may have courses in programming language design or implementation where it is necessary for them to obtain experience with the languages. Alternatively, they may be taking a course in algorithms and have frequent need to implement their solutions to problems. At this stage in these students' training it is important for them to be exposed to as many different programming situations as possible; the computer system should not give them a myopic view of their field of study.

Users from non-computer science courses

The members of this user group are learning both to utilize the computer as a tool in their respective fields by carrying out assignments which illustrate basic methods, and also to use library packages to aid them in their work. For example, students in this group may be in an econometrics or statistics class where they are performing sample regressions using library packages, or developing their own software to do this.

This group, being diverse, may yield many requirements. It is composed of future mathematicians, social and physical scientists and various kinds of applications programs are needed for their support. Thus, this group will also require a wide class of programming languages and a powerful library of applications programs.

Usage by instructors and staff *

These users typically have enormous time demands on them and thus require responsive, time-saving systems. It is essential for faculty to have the ability to develop pedagogical and applications libraries quickly and easily.

In terms of other requirements, those of this group may be correlated with the needs of their students. If the needs of the users in the groups described above are met, the needs of this user group will be met as well.

There are three main requirements in an educational computer system which can be distilled from the needs of the four user groups described above: (1) support for a wide variety of programming languages and applications libraries, (2) provision for quick program development and easy debugging, and (3) simple, sensible conventions.

It is important to note that *no* program execution speed requirements have been postulated. This is because repetitious and time-consuming program executions are not commonly found on the usual instructional university machine. While it is true that some students will utilize the computer to solve problems using library applications programs, the tasks tend to be small since the computer problems assigned in most courses are the minimal sized examples which demonstrate the desired principles or techniques.

Thus, system throughput is not critically dependent upon program execution speed. Rather, it depends on *program development efficiency* because constant program development is the activity which best characterizes the university instructional computer system.

ALTERNATIVE SYSTEMS

The needs of the various user groups may be met in different ways. The users may be supported on a variety of machines, each offering distinct facilities. For example, there may be the need for some special purpose computers to act as real-time laboratory aids or to be used stand-alone in an operating system course. However, the needs of most users will be supported either on non-interactive systems (batch) or on interactive time-sharing systems. It is to these systems that we next turn our attention.

Non-interactive systems

The non-interactive, or batch, machines are usually fast and capable of executing the largest programs. However, execution-speed/size efficiency is not obtained without a price. Batch machines are inefficient as a tool for program development, especially in an environment in which a large support staff is unavailable. In addition, they are unusable in situations where interactiveness is a requirement. Finally, in some sense, they are an improper model of a computer to present to the student, especially the neophyte.

As a program development tool for those learning to utilize computers, batch machines are very wasteful of both machine time and more importantly, human time. Associated with batch machines is the notorious *batch cycle*. This consists of the four step process which must be repeated numerous times while writing and debugging programs: (1) the program must be written or corrections made, (2) program (and job control cards) or corrections must be keypunched, (3) the card deck must be submitted and (4) the print-out must be awaited. This cycle is especially onerous in an instructional environment because students will make large numbers of mistakes, both conceptual and typographical. The latter mistakes are compounded by the fact that students *are not professional keypunchers*. Thus, the batch cycle will be repeated an excessive number of times even in the course of easy program development. This ties up both operator time, student time,

* The reader should note: that this paper is concerned only with instructional uses of computers in the university. If it is desirable to have the faculty and staff utilize the instructional machine for their own research purposes, the requirements can grow considerably.

and burdens the system with a constant influx of trivial tasks.

Further problems with program development are associated with the fact that debugging aids are necessarily static on a batch machine and therefore, they are not nearly as flexible as the dynamic aids on an interactive system. Again, this is an especially grave deficiency in a university environment due to the large number of errors that students will make.

In addition, the very fact that batch machines are non-interactive means that restrictions are placed on the flexibility of the system. Certain kinds of computer usage cannot occur at all and other kinds not very easily. For example, the batch machine is not usable in a laboratory situation where a student wishes to quickly analyze data necessary for an ongoing experiment. In other situations, it may be very desirable, though not strictly necessary, for the user to dynamically view the results of a program and be able to interact continuously with it.

The final disadvantage of batch systems is a psychological one. In using a batch machine, the student must necessarily learn of the computer as a static machine. The student sees the computer as a monolith which can talk only *at* him, never *to* him. Undoubtedly, this is a bad view to give to the student, especially the introductory student who may be uncomfortable with computers in the first place.

In summary, batch machines have the capability of executing large programs efficiently but provide an inadequate program development medium for students. They are restrictive in the kinds of tasks in which they are useful and are a poor model to present to neophytes. It should be noted, in light of the conclusions on the needs of the user groups, that the ability of batch systems to execute large programs efficiently is not extremely valuable on an instructional system. It is a capability which is not necessary for this environment.

Thus, the batch computing system does not fulfill the requirements of the user groups and is not a good choice as the basis of an educational computing system. This is not to say that batch machines do not have a place on campus. In the bulk of computer usage, efficient machine utilization is vital and batch machines can be used effectively. However, in contexts where there is almost constant program development and great diversity of needs, batch computers are not suitable.

There has been a variant on the batch machine which has given to the batch system some of the attributes of interactive systems. Systems like Wylber[1] have provided some degree of interactiveness and at the very least, allow the user to dispense with key-punching. To the extent that the system can provide interactive facilities, the system can meet the requirements described earlier.

Interactive systems

Although, interactive computing is an old concept in educational systems,[2] it is still rarely utilized.[3] The machines which are utilized for interactive computing are very diverse and encompass great differences in flexibility, machine cost, performance, and even interactiveness. Some systems have the capability of running in a batch emulation mode, while others provide no such opportunity. Some systems utilize only one language while others support many diverse languages. In this section, we classify the different types into four categories: (1) large scale time-sharing systems, (2) traditional mini-computer systems, (3) large mini-computer systems and (4) networked systems.

The large scale time-sharing systems are characterized by *many* simultaneous users, relatively large size and speed, and high flexiblity. In fact, the systems tend to be so large that many universities have too little computing to fully utilize a whole machine. Thus, the system must be shared with others. This can lead to high communication costs and other problems associated with an environment in which there is not complete control over the computing resource. However, as will be seen, from strict performance criteria, these machines do an excellent job of satisfying the educational requirements.

Since the large time-sharing machines are fast and possess large address spaces, they can run the most complex instructional programs. In addition, the large address space allows the use of many different language systems as well as the use of large flexible software designed to simplify the program development task. Furthermore, they can allow for any amount of interactiveness including on-line editing, dynamic program debugging, and graphics. Some systems even provide a batch mode in which more complex, time-consuming tasks can be run.

Of course, these machines do not have the efficiency advantages of the batch machines in raw execution power but as has been argued, this is wasteful in an educational (non-research) environment. In fact, it is argued here that even these large time-sharing machines, like the large batch machines, have built into them an amount of execution efficiency beyond that which is required in the instructional environment.

Traditional mini-computer systems have tried to provide the services of their large counterparts but with more restrictive environments and less powerful processors. Many systems have only a single language available, typically BASIC, and can handle between 5 and 15 users simultaneously. These machines have many of the advantages of the large time-sharing machines and thus satisfy some of the requirements of the user groups. However, they are not powerful enough to meet the flexibility requirements.

These machines are interactive and thus allow ease of debugging, instantaneous turn-around, and quick program type-in. However, due to both the restrictive-

ness of the operating systems which run on them, the speed of the CPU and the restricted main memory size, they cannot support languages powerful enough for many users. For example, the "BASIC only" systems are suited for some kinds of programming but certainly cannot fulfill the requirements for an intermediate course in computer science.

Thus, these machines do not fulfill the requirements of the user groups because of their low flexibility. They are not a good choice for a university system.

Large scale mini-computer systems are based on the great advances in hardware technology that have allowed for the creation of much larger and more powerful mini-computers. Although many of these machines have limited processing speed and memory restrictions, they are much more powerful than the mini-computers which were available only a few years ago. We contend that one or more of these machines are sufficiently powerful to provide for almost all of the educational computing needs for a university. Typically, each can serve between 20 and 50 users, each user having the ability to perform a wide variety of commands and utilize a wide variety of languages.

The reason that this can be achieved is that most requests for service in the instructional environment are trivial ones which have correspondingly little need for raw computing power. Thus, a relatively large number of users can be supported on a system of only moderate power. Occasionally, some students' requests will be beyond the power of the system. But, in the authors' experience, the number of such requests is typically small. For these students, alternative arrangements can be made.

An approach that may be used in the future is to connect a number of large-scale mini-computers directly to a powerful central processor in a hierarchical network. If any of the mini-computers become overloaded, they may send some users' requests to the more powerful machine to be executed. In this way, reasonable response time can be maintained on the small machines. In addition, the small number of users whose computations would normally overflow the large-scale mini-computer systems may be accommodated.[4]

Thus, of the time-sharing approaches, only the use of the more traditional mini-computer based system does not meet the requirements of the user groups. If the university administrator is concerned with performance alone, the large scale time-sharing system would be his choice. However, as the cost of such a system may exceed the funds available, it may be necessary to choose among the other alternatives.

COSTING THE VARIOUS SYSTEMS

The university administrator, after taking into consideration the merits of the various kinds of computer systems, as discussed in the previous section, must choose between systems based on their educational cost-effectiveness. This determination is exceptionally difficult[5] and is a two part analysis. First, the cost per standardized resource unit must be determined. Second, the educational benefit per such unit must be evaluated.

If educational effectiveness per unit resource is not considered, it is possible for a computer system to appear to be less expensive than it actually is. For example, it is easily conceivable (and even expected) that a student may require more runs on a batch system in order to debug and test a program than on an interactive system. Thus, even if the batch system were to have a smaller cost per job executed, the overall cost of program development could be higher.

The difficulties in determining cost per resource unit are also considerable. The construction of a standardized unit with which to compare systems is in itself a difficult task. This can be seen to be extremely hard when comparing batch and interactive systems. However, even on a single system, it is difficult to present clearly the assumptions that go into the determination of some costs.

Thus, whenever a definitive assessment of the benefits and costs of a given system is made, it should be kept in mind that the task is extraordinarily difficult and that the results of any studies should be viewed cautiously. Finally, explicit specification of the numerous assumptions used in any study are required if the study is to have any meaning.

The authors' experiences have been with educational time-sharing systems. However, it has become clear to them that even when comparing and attempting to describe the attributes of systems in this category alone, the number of items which must be considered is very great with respect to the description of the costs of a given system, and of its attributes.

For example, on interactive systems, it is common to use *connect cost per hour* as the yard-stick of cost. This is usually defined by the following fraction:

$$\frac{\text{Initial Cost/Yr.} + \text{Operating Cost/Yr.}}{\#\ \text{Connect Hours/Yr.}}$$

where

$$\text{Initial Cost/Yr.} = \frac{\text{Hardware} + \text{Software Cost}}{\#\ \text{Years of Amortization}}$$

But, it should be immediately seen that this fraction can vary substantially, perhaps by an order of magnitude depending upon the interpretation of the various terms. For example, are actual yearly connect hours utilized, or some "reasonable" figure which could be assumed to be the maximum (or minimum) number available? Or is the "reasonable" number of connect hours based on raw hardware limitations or upon the maximum number of users which can obtain some reasonable response time? There are, of course, many more questions which could be asked.

It should also be noted that the cost from university to university can vary greatly. Perhaps, the university can support some percentage of the cost of the instruc-

tional machine by utilizing it for some research or some grants can be obtained to cover certain parts of its development. Also, there can be wide variations in the cost of the hardware from institution to institution.

Because of these difficulties, no attempt is made to provide a detailed cost comparison of various time-sharing systems. Unfortunately, this would be the only way to demonstrate that large mini-computer based systems are cost effective. But this would be difficult, in general, since there will be instances where circumstances particular to a given institution make a different kind of system more cost-effective. Instead, a detailed case study of the Harvard undergraduate time-sharing system is presented.[6,7] This system is one in which the authors have extensive experience and one which has been running long enough to provide comprehensive data. In the particular situation in which it is used, it appears to provide for extremely inexpensive instructional computing. It also appears likely that there might be many comparable situations.

A LARGE MINI-COMPUTER BASED SYSTEM EXAMINED

The present Harvard undergraduate time-sharing system (HRSTS) is utilized in the support of the computing requirements of most undergraduate courses, some graduate courses and a limited amount of research usage. Among others, courses in applied mathematics, economics, mathematics, engineering, music, and even Arabic utilize the system. The hardware used is a Digital Equipment Corporation PDP11/45 processor,[8] 240K bytes of core memory, a fixed head disk for swapping, 116 Megabytes of on-line disk storage for files, a high speed printer, card reader, paper tape reader/punch, 2 DEC-tape drives and about 28 terminal ports.

The operating system used is a modified version of Bell Laboratory's UNIX system.[9] The language processors currently in use on the system are 2 assemblers, 2 text editors, numerous high level languages including BASIC, FORTRAN, C,[10] PPL,[11] ECL,[12] and LISP as well as a wide variety of utility programs. The user command interpreter was specially designed for ease of use based on past experience with other time-sharing systems.

The system is operational 23 hours a day, 7 days a week with periodic maintenance disturbing this schedule somewhat. The load conditions vary from semester to semester. During the first semester, PPL is used heavily by the 400 students of Harvard's general introductory computer course. During the second semester, the assembler and LISP are most heavily used due to an intensive introductory computer course for more advanced students. During both semesters, FORTRAN and BASIC are utilized extensively by students dealing with numerical methods and due to continual system development, the language C is also highly utilized.

The system is operated by one full-time administrator and numerous "terminal watchers." The latter are students who are hired by the university to act as combination operators, programming assistants, and systems programmers. During most of the day, they are available. This is especially valuable for the large number of beginning users.

The use of terminal watchers highlights one of the cost advantages of the university owning its own system. The cost of the terminal watchers is low due to the lower cost of student wages and the fact that a payment from the university to a student is really an intra-university transfer payment which is not too costly to the university and very beneficial to the student.

The present operating costs for the system are sketched in Table I. It should be noted that the budget is inflated by four items: (1) the cost of the terminal watchers, many of whom would not be necessary if it were desirable to have only operators on duty during usual hours or if such personnel could be recruited from the teaching staffs of the courses using the system, (2) the high cost of having rented, high speed video display terminals, (3) the high cost of maintenance contracts with the various manufacturers (as opposed to in-house maintenance), and (4) the cost of having dial-up lines on the system. The latter cost shows that communication costs are a very important consideration in the overall costs of a time-sharing system. In a different environment, the costs could be reduced significantly.

The capital costs for the system have been amortized over a three year basis and come to approximately $50,000 per year. The amortization period is not based on the life expectancy of the system for the system is now 18 months old and there is every expectation that it will last considerably longer than an additional 18 months.

The system usage, over the Spring and Fall semesters of 1975, is shown in Table II. As can be seen, the total number of connect hours utilized during the year is 43,500. It is the belief of most people associated with the operation of that system that this number of connect hours is close to the maximum that the system can support.

Based on the *actual* connect hours of the year 1975, the connect cost per hour is approximately $3.20. It

TABLE I—Approximate Operating Costs for HRSTS

Item/Description	Amount
Terminal Watchers & Other Salaries	$41000.
Supplies	$ 5000.
Telecommunications	$ 9000.
Terminal Rental	$22000.
Maintenance	$13000.
Total	$90000.

TABLE II—Total Connect Hour Usage of HRSTS during 1975 by major usage category

Category	Spring	Summer	Fall*
Applied Mathematics	7685	—	300
Arabic	99	149	400
Biology	232	110	1100
Chemistry	1492	215	100
Economics	29	55	200
Engineering	496	115	300
Grad. School of Design	1028	10	1500
Independent Usage	1053	488	1000
Mathematics	753	93	300
Natural Sciences	297	28	8100
Physics	1159	575	300
Social Sciences	346	90	100
Statistics	362	11	500
Research	336	135	200
Sys Work/Term. Watchers	3288	1995	4000
Other	450	1270	800
Totals	19105	5339	19200

* projected based on Dec. 20, 1975 totals.

should be noted that this figure does not include an extraordinary once-only development cost of $50,000 for some of the system software. If the reader is concerned with duplication cost of the system, this figure should not be taken into account since most of the software is now freely available to other educational institutions.

HRSTS COSTS IN PERSPECTIVE

The determination of the relation of these costs at Harvard to real costs in other situations must necessarily take into account the following consideration.

(1) *Usage*—The system at Harvard is highly used, often in the very early hours of the morning. The impact of this is substantial on the connect cost per hour figure cited above.

(2) *Communication Costs*—At Harvard, there are many dial-up lines available. If the terminals are in one or more centralized locations on campus, the number of dial-up lines can be reduced substantially.

(3) *Overhead*—The costs given do not include the cost of utilities, room space nor the *de facto* use of some of the administrative structure of the Harvard University Science Center.

(4) *Operator Attendance*—The cost of operator coverage could vary substantially at other installations depending upon the availability of students to handle this function and the amount of coverage desired. It could conceivably be made the responsibility of teaching assistants.

(5) *Peripheral Hardware*—Considerable peripheral hardware exists at Harvard that might not be needed elsewhere. However, additional items might be necessary if real-time or graphics applications are desired.

(6) *Application Programming Costs*—These are specifically not included in the connect cost per hour figure cited above except to the extent that they are handled by the terminal watchers. In some situations, it might make sense to include these costs.

(7) *Amortization*—The period at Harvard is three years. It might be more reasonably set at five to seven years.

(8) *Specialized System Software*—Most of the software used at Harvard may be obtained free by other universities. However, the development of new software might be required for certain situations.

(9) *Hardware Advances*—The compatible Digital Equipment Corporation PDP11/70 processor[13] can support at least twice as many users without very great increases in cost.

It is possible to reduce the cost per connect hour to $0.50 per connect hour under favorable circumstances (no dial-up lines, operator coverage by teaching assistants, local maintenance, longer amortization, and purchased low cost terminals). Since most environments will differ somewhat from this extreme, the expected costs will be in the $1.00 to $3.00 range.

CONCLUSIONS

The low cost of HRSTS lies in the fact that the hardware is tailored to the needs of the users. Such systems utilizing large mini-computer systems, tend to be just powerful enough to support the vast majority of the users' demands. In the future, it is conceivable that a hierarchical network based on machines of differing power will be the logical extension of this. Easy computing tasks will be serviced by the simplest and lowest cost machines with the more diffcult tasks being passed on to larger machines.

This division will allow the cheapest machines to be utilized in most instances and the more complex and expensive machines will be saved for the limited amount of overflow. This approach will allow the benefits of the largest time-sharing systems to be coupled with the very low costs of the mini-computer based systems.

Today, a large mini-computer based system, not unlike Harvard's, can offer very low cost, flexible computing that satisfies most requirements for an educational system. The services provided by such systems are much more suited to the needs of the university user community than are those provided by batch or single language systems. Although not as powerful as the large scale time-sharing system, they are sufficiently powerful to satisfy the needs of most users. Furthermore, such systems can be run with very low cost, with connect costs in the $0.50 to $3.00 range.

REFERENCES

1. Fajman, R., J. Borzgelt: "Wylber: An Interactive Text Editing and Remote Job Entry System," *CACM*, Vol. 16, No. 5, May, 1973.

2. Kemeny, John G. and Thomas E. Kurtz, "Dartmouth Time-Sharing," *Science*, Vol. 162, No. 3850, October 11, 1968.
3. McCracken, John, "Is There a FORTRAN in Your Future," *Datamation*, May, 1973.
4. Prenner, Charles J. and J. P. Buzen, *Proposal For Research on Hierarchical Computing*, Ctr. for Res. in Computing Tech., Harvard University.
5. Austin, John E., *Costing Computer Aided Instruction*, Ibid.
6. *HRSTS Terminal Users Guide*, Harvard University Science Center, August 1975.
7. Prenner, Charles J., Using Unix in an Instructional Environment, *Proc. COMPCON* 1976.
8. *PDP11/45 Processor Handbook*, Digital Equipment Corporation, 1973.
9. Ritchie, Dennis M. and Ken Thompson, "The Unix Time-Sharing System," *CACM*, Vol. 17, No. 7, July 1974.
10. Ritchie, Dennis M., *C Reference Manual*, Bell Laboratories.
11. Taft, Edward A. and Thomas A. Standish, *PPL User's Manual*, Ctr. for Res. in Computing Tech., Harvard University, TR 21-74.
12. Wegbreit, Ben E., "The ECL Programming System," *Proc. FJCC* 1971.
13. *PDP11/70 Processor Handbook*, Digital Equipment Corporation, 1975.