reprinted from

COMPUTER
magazine

Computing in Higher Education: The Athena Experience

Ed Balkovich, Digital Equipment Corporation

Steven Lerman, Massachusetts Institute of Technology

Richard P. Parmelee, IBM

Computing in Higher Education: The Athena Experience

Ed Balkovich, Digital Equipment Corporation
Steven Lerman, Massachusetts Institute of Technology
Richard P. Parmelee, IBM

Project Athena at MIT is an experiment to explore the potential uses of advanced computer technology in the university curriculum. About 60 different educational development projects, spanning virtually all of MIT's academic departments, are already in progress.

Computing has been widely discussed as having the potential to radically change higher education.[1] The advent of low-cost, high-performance personal computers has been touted by some as the savior of the American university system and condemned by others as the beginning (or even the final stage) of its downfall.

With the veritable blitz of media attention that has been focused on the educational uses of personal computing over the past few years, it has become increasingly difficult to realistically assess the actual changes this new technology is likely to accomplish. Part of this difficulty has resulted from the largely technology-driven frame of reference in which computation has been viewed; universities have often simply accepted the specific technologies of current hardware and software before asking how they might use them.

In this article the use of computation in higher education is approached from an entirely different perspective. We begin by speculating on how computer technology—in its broad sense—might be of actual use in the curriculum, and we do not overly concern ourselves with the question of whether those uses are practical within the context of current technology. In particular, we seek to identify areas where current educational methods have observable deficiencies that might be alleviated by the use of appropriate software/hardware combinations.

After presenting some pedagogically useful paradigms concerning the use of computation in the curriculum, we then proceed to develop a model of computation that could sustain those potential uses. Finally, we examine how the wide-scale implementation of this model might in turn influence the university as an institution.

The ideas presented here are derived from our work on Project Athena at the Massachusetts Institute of Technology (MIT). Introduced in May 1983, Project Athena is an experiment to explore the potential uses of advanced computer technology in the university curriculum. MIT has enlisted the assistance of the Digital Equipment Corporation,

[1] See *The President's Report, 1983–1984*, by D. Bok, Harvard University, Cambridge, Massachusetts, March 1985, for a recent and cogent contribution to this debate.

ACM/IEEE-CS Joint Issue © 1985 ACM 0001-0782/85/0110-0112/$00.75

the International Business Machine Corporation, and other corporate sponsors, each working independently with MIT. These corporate sponsors are providing (as grants to MIT) the hardware, software, equipment maintenance, and some of the technical staff. The university itself is raising $20 million to support the needed software, operations, and staff. By the end of the project, MIT will have installed a network of about 2000 high-performance graphics workstations. More importantly, about half of MIT's $20 million budget has been made available to faculty members for the creation of new-applications software to be used in MIT's own curriculum. About 60 different educational development projects—spanning virtually all of MIT's academic departments and collectively offering a wealth of ideas about how computation can be used in the curriculum—are already in progress under the auspices of Project Athena.[2]

Potential educational uses of computation

In an academic setting, computer systems could probably be used in five distinct ways. First, the original use of computers as an integral part of teaching and research in computer science and electrical engineering will undoubtably continue. Second, the current administrative use of computers for financial management and other record keeping will also remain. Third, the fairly recent availability of generic software for text processing, spreadsheet analysis, personal database management, etc., has already become widespread within universities. Fourth, the use of computers for electronic mail and related forms of asynchronous communication is still in its early stages. And, fifth, what will probably be the most controversial use is that of the computer as an integral part of the instructional process. Focusing on the fifth point, we will explore in this section some possible future academic uses of computing.

The computer as a simulator of complex systems. One of the most common requests in faculty members' proposals to Project Athena is to use the computer as a simulator for physical or social phenomena that are too complex to be understood from mathematical formulations. The subject areas involved are as diverse as orbital mechanics, supply/demand equilibriums in economic systems, turbulent flows of fluids, electromagnetic fields around charged objects, and the analysis of policies on releasing water from dams into major river basins.

In each of these areas, students have usually learned mathematical formulations that model the phenomena under study. Many students, however, find it difficult to translate the abstract, mathematical formulation into a real, intuitive understanding that they can internalize and use in subsequent courses. Although all students should understand the formal mathematics, not all can exercise the model on problems of meaningful size, either because of the computational effort or because the answers are too hard to portray. In many cases, faculty confine examples to problems that are "solvable" on the blackboard, and students' assignments are limited to similar, often trivial, examples. Appropriately constructed software—running on hardware with the capacity to solve and to portray larger scale (and consequently more realistic) applications of the mathematical formulations—allows students to test their intuition against the predictions of formal models. The student can be asked to pose a variety of "What if?" questions about problems so sufficiently rich that not even the qualitative answers may be obvious. A measure of the success of the user interface and presentation software, if not of the underlying simulation code itself, is the extent to which the student is drawn into this "What if?" sequence. It should be fun! To the extent that this is successful, it brings to the curricula a powerful new mode of learning.

The obvious hazard of simulation is generally that the chances are increased that the results presented may be artifacts of flaws in the model, in its evaluation, or even in the presentation itself. Making simulation more central to the educational process carries as a concomitant the requirement to give it some greater emphasis in the curricula. Simulation is now emerging as a potential form of scientific inquiry, paralleling the two more traditional modes of theory and experimentation. If this admittedly controversial trend continues, the computer will undoubtably become a central element for teaching students how to devise simulations as part of the scientific method.

The computer as a laboratory instrument. The creation of meaningful laboratory experiences, particularly at the undergraduate level, has often been a frustrating task for universities. High-quality laboratories are expensive, and many universities have been unable to recapitalize them as technology has advanced.

[2]See *Faculty/Student Projects*, Project Athena, MIT, Cambridge, Massachusetts, 1985, for more detailed descriptions of these projects.

Most students can recall the disappointing experience of recording data in a lab subject, only to find that the results bore virtually no relationship to the theory taught in class. The cause of this frustration with laboratory experiences may go back to the unrealistic way in which we have gone about creating such courses. First, real insight from experimentation is rarely derived from a single experiment—students should have a wide range of experimental data available when seeking to draw inferences or to validate a theory. Second, the student in the lab often does not have the opportunity to correct flaws in experimental technique because the data are acquired in a single, brief lab session that does not provide an immediate means of checking data as they are collected. Also, the particular theory that the lab is intended to illustrate is often too disconnected (in the logical rather than physical sense) from the experiment the student actually conducts. Finally, experiments that yield truly significant results often require data to be subjected to sophisticated analysis techniques in order to reduce data, filter noise, etc., before meaningful inferences can be drawn.

Instead of the traditional student labs, let us consider an alternative in which data are acquired digitally by a computer. The computer allows the data to be displayed, compared directly with predictions from theory, postprocessed, stored for subsequent analysis, shared with data from students running the same or related experiments, and compared with data acquired from more sophisticated experimental apparatus. The most obvious benefit of this approach is that many of the unnecessarily tedious tasks associated with doing the lab exercise can be eliminated, allowing the student to focus on the physical phenomena under study. More subtle is the possibility for students to run their own experiments, with data being subject to analysis and comparison with standardized results right in the lab. Students could be asked to analyze not only their own experimental results (which are often too limited for meaningful inferences to be drawn), but also the data of other students' experiments, which would collectively span a wider range of interesting conditions. The predictions of alternative theories taught in class could be directly compared with experimental results, permitting students more insight into flaws in the design of the experiment and its relationship to the application of a particular theory. Thus, the range of validity of a particular model or experiment could be readily explored. With the ability to move data easily from lab, to classroom, to residence—an integral part of such a system—more careful and detailed analysis and reporting would result.

A potential hazard of using the computer as a virtual laboratory is that it could displace actual experimentation.

This type of laboratory experience would have the potential of shifting lab subjects away from an emphasis on obsolete experimental techniques to one where the spirit of scientific discovery is learned.

The computer as a virtual laboratory. The experimental laboratory remains an important pedagogical device for allowing students to explore the relationship between theory and experimentation, but there are many fields of inquiry where those same benefits cannot be gained at all. For example, experiments on nuclear-power-plant operations are obviously too hazardous to allow students to discover key operational trade-offs through direct experience, and many situations in physics involve phenomena that occur only at velocities approaching the speed of light. In such instances, scientists have traditionally invoked the concept of a "gedanken" experiment—to have students think through a conceptually feasible but physically impractical experiment. With the aid of appropriately constructed software, such "experiments" would become less abstract, and students would gain more from the experimental situations.

The distinction between "virtual laboratory" and "simulator" paradigms rests on how the programs are intended to be used. For example, a program that evaluates population levels of certain insects as a function of climate, insecticides, etc., may lie at the heart of both a simulator and a virtual laboratory. The simulator covers this program with a layer of software that permits, even encourages, the student to change and redisplay variables. The virtual laboratory uses a layer that encourages, or even requires, systematic identification of key parameters, their range and sensitivity.

A potential hazard of the use of the computer as a virtual laboratory is that it could displace actual experimentation in the curriculum. In our view, this would represent a serious misuse of computation. A virtual laboratory, however, may make actual lab experiences more productive. For example, students might review laboratory techniques in a virtual laboratory first, so that they would be better prepared for the actual lab experiments later.

The computer as a tutor. Most of the early applications of computers in education (beyond using them to teach about computers themselves) focused on the computer as a teacher. Programs that presented material, asked questions, and branched appropriately were by far the most common educational applications, and the experience with such programs was not particularly encouraging. However, recent advances in artificial intelligence have opened up the possibility of using computers as "expert tutors." The key distinction between this form of software and early computer-aided instruction (CAI) is that in an expert tutor the student remains the primary agent in the student-computer dialogue. With the early CAI software, the student tended to be passive, with the computer controlling much of the interaction. In an expert tutor, the student acts as the primary problem solver, and the computer offers advice on request.

Consider as an example an Athena project in civil engineering that is building a tutoring program for introductory-level structural analysis courses. The program develops typical problems for students to solve, but the student is then left to formulate an appropriate set of equations describing the equilibrium of forces and moments. The software can analyze the system of equations and diagnose common errors on request. For example, students often forget to impose a convention on algebraic signs of forces to represent whether they are up or down. On request, the program can offer increasingly detailed comments about what is wrong with the student's proposed solution.

The creation of this type of program requires much more sophistication than did the early CAI software. With expert-tutor software, a general inference capability for the class of problems under study must be developed, and a careful study of common errors made by students has to be undertaken. The software needs to work with the mathematical symbols that are required to represent the subject under study. In the case of engineering and science, this will often require algebraic manipulation of mathematical expressions. Subjects such as foreign languages will require computer interfaces that can cope with natural languages.[3] However, there are still unresolved questions about how to best construct expert-tutor software and whether the state of the art in knowledge-based systems is at the point where useful expert tutors can be created for complicated subject areas.

A critical advantage of computer-generated images over static textbook pictures is that students can manipulate them.

The computer as a textbook. In engineering and the sciences, there are often established conventions for graphical representation of information. For example, in chemistry there is a well-defined notational system used to display the composition and geometry of molecules. Textbooks rely on these abstractions to display information about two- and three-dimensional geometry. However, for many students, the translation of these iconic pictures into the mental models of the actual objects they are intended to represent can be difficult. A critical advantage of computer-generated images over static textbook pictures is that students can manipulate them—that is, rotate, compress, or edit a particular display. Such graphical tools place the students in control of the representation of material and free them from being locked into a single set of predefined pictures.

The importance of allowing students to control and manipulate graphical representations is perhaps most clear in computer-aided design (CAD) applications. The need for architects and engineers to rearrange components of designs easily has been crucial to the success of CAD in industry, and it is reasonable to suppose that the same benefits would extend to learning the design process itself. Another advantage lies in the use of animation, particularly in helping students visualize what happens in time-dependent systems involving moving objects, vibration, or the evolution of complex social systems (such as the national economy). Finally, the order in which the components of a picture are drawn can often provide students with as much insight as does the picture itself. For example, when displaying the magnetic field around a set of charged objects, it is important to display lines of symmetry and to exploit such symmetry in the solution. Computer-generated displays that first draw in one quadrant and then reflect that drawing into a symmetric quadrant, do more than illustrate the field under study; they also help the student gain insight into more general problem-solving methods.

The computer as a blackboard. Most of the limitations of textbooks

[3]MACSYMA, a program developed by Joel Moses at MIT, illustrates that software to support mathematical manipulation can in fact be constructed. Progress in natural-language processing has been far slower. Within Project Athena, work is underway on the development of software capable of natural-language recognition in five foreign languages at the level of a first- or second-year university student.

apply to blackboards as well. Although students do benefit from watching how images are constructed on a blackboard, they may also have to bear the burden of the instructor's limited artistic abilities. Moreover, many complex images are very difficult to draw, and once displayed on the blackboard, they are difficult to change in a clear way. Large-screen, high-resolution projection is a significantly better medium, particularly in large lectures. In addition, the computer-generated graphics used in class can easily be made available to students for further study outside of class.

The computer as a special-purpose learning environment. Certain academic disciplines require resources that, for most students, are unavailable. For example, student composers of orchestral music ideally should (and rarely do) have access to a complete symphony orchestra to hear their own compositions. The computer, however, can supply the foundation for a specialized environment that provides for input on a clavier keyboard and output from sound-synthesis hardware with high-resolution graphics to display musical notation. This form of "composer's workbench" would offer the student a powerful tool for exploring music and would provide a qualitatively different experience from more conventional approaches.

The computer as a communications medium. Universities are, to a large degree, in the communications business. Written reports, if not central to a technical curriculum, are important. The lab report is the student's prose description of the theory (including math symbols) and the lab experience (including graphs and charts). Computer-supported processing of intermixed text, symbols, and graphics, though possible, is still too awkward for student use for project reports. Thus the current handwritten, cut-and-paste approach necessary to prepare a lab report is a high barrier to "What if?" revisions. The payoff for text-processing support for these technical reports will be at least as high as that already observed for simple text. The result is not just better reports for equal or less effort; it is that greater academic emphasis can be placed on the quality of the report itself, allowing it a more central place in the curricula.

With personal-computer access, the library can enhance some of its present services and provide entirely new ones as well.

The role of the library in the curricula will also shift. Currently the library is viewed as a place rather than as a set of services. Students go there, when it is open, to search the literature. With personal-computer access, the library can enhance some of its present services and provide entirely new ones as well. An Athena project by the library to allow computerized access to its catalogs is the first step in this direction. As more material becomes machine readable (for example, videodisc storage of whole journals can be on line), students will not only search for relevant material, but will obtain copies as well. By making the library more accessible, scholarship and the role of the library in the curricula can receive greater emphasis.

In addition to making libraries more accessible, computers will deliver entirely new services. Personalized newspapers culled from news services are already available at MIT. An Athena-supported project to provide campuswide access to the course catalog and to students' evaluations of courses will certainly evolve into a rich environment containing course outlines and sample problem sets. Finally electronic mail and bulletin boards will help develop a sense of community.

The computer as a mediator. Linked computers would offer the possibility of using multiplayer games to teach in subject areas (such as economics, logistics, and political science) where the true complexity of the system under study arises from the interactions of numerous individuals. Such computer-mediated simulations have been used as either totally self-contained activities or as a preliminary procedure to a further classroom-based exercise.

Obviously, computer-based communication in mediated games has a quality different from direct, face-to-face contact. In some situations this makes computer-mediated gaming a somewhat artificial experience. However, there are instances where it is important to be able to exert control over the information that can be transmitted to actors in a game. For example, students can be taught how changes in the ability of mock superpowers to communicate during hypothetical crises can affect likely outcomes. By regulating how often information can be exchanged or by altering the allowed content of messages, various possible forms of communication can be simulated. An analogous situation exists in business simulations where legal restrictions that may affect market outcomes are imposed on discussions between competing businesses.

The computer as recreation. It would be incomplete to think of

computers at a university only in terms of their use in the curriculum. For many students, the computers provide a rich and varied source of extracurricular activities. In addition to games, computers support the activities of many campus clubs and individuals. For example, a club at MIT is using Athena's computation resources to develop a simulated mission of the space shuttle. Another club, the Student Information Processing Board, has been given access to the UNIX source code and a machine to experiment with, and routinely makes significant modifications in both the kernel and utilities.

The computational environment for education

Now that we have examined some possible future academic uses of computing, we will turn our discussion to the computing environment that would be needed to sustain these uses. Our point of view is a practical one. Although the cost of some components of campus computing systems is expected to decline, computing will never be free. This is particularly apparent when some of the indirect costs of computing—such as the faculty and student time required to develop, maintain, and administer academic software—are taken into consideration.

We believe that the envisioned computing environment can be realized in a few years. At the moment, however, MIT and the vendors involved in Project Athena are subsidizing the resources needed to experiment with the computing model. We do expect that the computing environment will be self-sustaining by the end of the experiment.

Assumptions. In projecting future computing needs, we have assumed that computing will be available for use in a variety of courses spanning many disciplines and that it will be incorporated into many settings on a campus. We also expect that the strategies for capitalizing computing will be flexible. They could range from personal computers purchased by individual members of the academic community to computing plants owned entirely by the institution. MIT, like many other campuses, will most likely use computers acquired both by individuals and by the university. In any case, academic computing must be cost-effective in order to sustain itself; it has to be designed and delivered in a way that takes into account the "true" costs of development and operation. Finally, computing needs to preserve individual and institutional investments in the presence of rapidly changing technology.

Implications. *Ubiquity.* If computing is to become an integral part of the educational landscape, it must be treated as a utility. A time-sharing system whose response degrades in midafternoon must be regarded as a failure mode of a public utility in much the same way that a "brownout" is viewed as a failure mode of an electrical utility. Campus computing facilities need to be predictable. We believe that this can be achieved by systems that dedicate a computer(s) to one user for a session.

Time-sharing systems do not scale to support an entire campus. Even networks of time-sharing systems fail to provide predictable responses for computing environments that emphasize computationally intensive applications or interactive graphics. Because personal computers lack the information-sharing and communications features of time-sharing systems, they are an inadequate solution to the problem unless they are set up as part of a larger computer network supporting information sharing. Of course, there must be a sufficient number of individual computers available in order to avoid shifting queueing delays from the CPU of a time-sharing system to the keyboards of a network-based system. What we do not yet know is how many workstations are needed to support extensive educational uses of computing.

Ownership. While future educational computer support will most likely be provided by a mixture of individually owned and institutionally owned computers, institutions will own and operate communications networks that link private and public computers. The acquisition and operation of such networks are rapidly becoming the responsibility of traditional computing centers. The data network will be a basic utility of the campus, and like any other utility, it may require its subscribers to pay a one-time installation charge or a recurring service charge.

Computers connected to the network can be divided into two categories: service computers and workstations. Service computers will provide generic services (e.g., filing or authentication) to one or more client computers, and they may include special-purpose or high-performance hardware. This category of computing will be owned and operated by institutions. Labor-intensive services (e.g., archival storage or a highly reliable filing system) and services involving consumables (e.g., a print service) will probably be offered on a fee-for-service basis. This style of operation will emphasize the need for authentication in network-based educational computing systems.

Students are mobile and education takes place in many locations on a university campus, and although they will use computers in a variety of settings and for a variety of reasons, we do not envision students making extensive use of portable computers. Portable computers do not now have, and are not likely to have, certain features needed for future educational applications (e.g., high-resolution bit-mapped graphics). We also expect that computers will be installed in specialized facilities (e.g., laboratory computers) and that students and faculty will be using one or more workstations in their day-to-day routine. If the training required for the operation of different workstations varies significantly, it could be a barrier to their effective use.

Both individuals and institutions will most likely own workstations, and in each case, a workstation will be operated by one individual per session. Whereas privately owned workstations will be located in residences or in offices, institutionally owned workstations will be located in the public facilities of universities (e.g., laboratories, classrooms, libraries, and other public work areas) and will be available to any legitimate user.

Cost-effectiveness. The cost-effectiveness of educational computing must be judged from several standpoints. Computing will continue to add value to education, but the acquisition of the computing power needed to improve the quality of education will require significant and direct institutional and personal investments.

There are equally large indirect costs. For example, the training required to use computing in a subject (e.g., foreign languages) represents an educational overhead. If computing is used extensively, universities will have to limit the amount of training time. If not, the training costs could reduce the amount of time available for presenting course materials. Thus, this indirect cost must be carefully managed so that it does not detract from the primary purpose of education.

In the course of a four-year education, significant improvements in computer technology are likely to occur. Student ownership of computers is one way for universities to continually upgrade their computing facilities with each freshman class. However, this too must be carefully managed to ensure an adequate spectrum of facilities to meet educational objectives and to avoid introducing serious incompatibilities that would render older systems unusable.

Since educational software represents a tremendous institutional investment that must be preserved in the presence of technological change, institutions must make certain that the software they create and acquire can tolerate changes in technology. The technology available at a given price will improve with time, and new students will naturally wish to buy the most current technology. It should also be possible to move existing educational software to new hardware with minimal effort.

Different types of hardware may be provided by a single vendor—representing valid price–performance trade-offs. For example, student-owned computers may be significantly different from the university-owned computers used in a CAD laboratory, and educational software must be able to tolerate such differences. Ideally, it should be possible for students at other workstations to use (in a limited fashion) materials prepared in a CAD lab. For example, a student might use a private workstation to prepare a lab report that would include graphics generated earlier in the lab.

Sometimes the cultural orientation or business policies of individual institutions may lead to multivendor solutions. Computing supplied by multiple vendors presents universities with the same challenges to their investment in educational software as do diverse workstations owned by both students and the university.

These observations suggest that, in order to be cost-effective, educational uses of computing must anticipate changing technology. Applications have to be built on higher level abstractions that hide the underlying technology. Ideally, these abstractions evolve slowly and can be reimplemented on new technological bases to preserve existing educational software. Such abstractions can be provided by a single vendor—for example, through a well-integrated set of operating systems, communications protocols, and layered products that span the vendor's hardware products. Alternatively, the abstractions might be generic—for example, an operating system, network protocols, and layered products that span multiple vendors' products. If every user interface to educational computing presented a computing environment based on the same set of abstractions, training costs could be reduced. In addition, if the choice of abstractions were to be widely accepted by other universities, this approach would also encourage the exchange of educational software.

Athena's computational environment

The first two years of Project Athena were supported by a campus-wide network of time-shared computers. This system was de-

signed to be deployed quickly and used to support the initial development of academic software by faculty and students. It simulated many of the features we expected to include in the computing environment provided by workstations.

We are now entering the second phase of the project, which is adding public and private workstations to the network and which is converting existing time-shared computers to service facilities. The conversion should result in a system with the following general features.

The system. Athena will provide its users with a system of individual computers linked by a network. Users may own a computer (private workstation) or may use public facilities (public workstation). All workstations will be dedicated to one individual per session and will be designed to deliver the computing power needed for interaction and graphics. The network will offer access to other computers operated by MIT and will provide services to all workstations.

The owner of a private workstation may assume that the machine is never used by anyone else. A public workstation will be dedicated to a single individual per session; however, users will need to assume that others use it as well. Public workstations are "serially reusable" resources, and it will be possible for a user to "reset" a public workstation to a known state and to easily reconstruct information "cached" in its local mass storage.

Members of the campus community will be able to use any public workstation or privately owned equipment, and they will not be limited to just one computer. The system will provide facilities that will make it possible to share information and to access data and programs from any computer. For example, service computers will provide access to systems libraries (containing system software needed to reconstruct information cached in a workstation's local mass storage), to class libraries (containing course-specific data and programs), and to user lockers (containing user-specific programs and data).

The characteristics of workstations are motivated by their applications, and Project Athena assumes that advanced workstation technology will enable significant advances in the application of computers to education. It also assumes that the cost of such advanced technology will be reduced with time. The hardware used in the project is limited to a few types of computers and is provided by multiple vendors. The general features of an Athena workstation include

- a memory from 1 to 16 Mbytes with memory-management features that can be used to communicate among independent address spaces and to effect dynamic linking in a large address space (required to construct software such as LISP-based tutor);
- a high-performance processor capable of executing at least one million instructions per second (to explore realistic problems requiring complex calculations);
- a high-bandwidth, bit-mapped display (to provide high-quality interactive graphics);
- an "open" design that can be extended to include peripherals such as keyboards, sound-generation devices, video input/output, data acquisition, and control devices (required by specific laboratory applications and special-purpose environments such as a composer's workbench).

The network, which is representative of networks being developed at other universities, is implemented with multiple technologies and is based on a high-speed backbone network. Existing campus facilities have access to high-speed, local-area networks connected to the backbone network via special-purpose gateways. Future facilities are likely to use both high-speed, local-area networks and lower speed, wide-area networks connected to the backbone network.

Coherence. Project Athena must minimize unnecessary differences and limit training costs in order to sustain itself. Academic applications need to share common code and build on one another. It ought to be possible to make only a one-time investment in training that is applicable to all uses of the system or to make the use of applications and system software self-evident. It must also be possible to run most applications and to access information using any public or private workstation.

A key software module is the operating system, which insulates the user and applications from changes in technology and differences in the hardware supplied by multiple vendors. Other major software modules include discipline-specific software (e.g., software that illustrates electromagnetic fields) and general-purpose software (e.g., word processing). These modules define three interfaces that must be coherent:

- system interface—the interface between the operating system and applications;
- application interface—the interface between applications;
- user interface—the interface between the user and the applications and operating system.

These coherent interfaces provide the base of abstractions needed to preserve the investment in educational software and limit training costs.

The computing environment, which includes operating-system functions and general-purpose applications and libraries, is the same on all workstations. Project Athena uses a single operating system and a single communication protocol family to implement its computing environment. The operating system is UNIX, which provides the foundation needed to port all applications to all types of workstations. In addition, the same suite of general-purpose software (e.g., an editor, a word processor, a graphics library, etc.) is available at every workstation. UNIX hides the underlying computer technology from the user, limits the training costs, and provides a framework for writing (ex)portable applications that can be made to work with different types of workstations. At the moment, the training required to use UNIX and the general-purpose software is significant. However, it is a one-time investment that we believe will be reduced with experience and with refinements to the system.

The TCP/IP protocols are used to implement the same "virtual network" for all network technologies used in the project. The virtual network hides technology-specific features. A uniform virtual network is needed to provide common communication functions that span multiple vendors' computers. With careful planning, the TCP/IP protocols could be replaced by another protocol suite with similar functions.

The investment required for the development of new academic applications can be reduced in several ways. One approach is to look for "vehicles" other than one-of-a-kind programs for delivering educational materials. Some examples of existing vehicles are spreadsheets or application generators for drill and practice materials. Another approach is to support methods and techniques that would reduce the effort required to create new software. An example of this would be a library of routines for 3-D graphics. The former approach requires considerable innovation and is best left to exploration by faculty and students; the latter can be systematically attacked by defining and implementing a coherent interface between applications.

A working hypothesis of Project Athena is that most scientific and engineering applications can usefully interact only when employing a small number (20–30) of data types (e.g., graphs, arrays, tables). The application interface can be defined by implementing common representations of these data types and mechanisms for manipulating them. This motivates our goal of keeping the number of "supported" programming languages small because it is practical to provide such implementations for only a small number of languages.

If the number of ideas in the application interface is small and concise, then the programming interface of an application is easier to understand. If an application's use of the concepts in the interface is complete, then programs can be combined in standard ways. A program that takes advantage of such an interface is more likely to be used with other programs. For example, a word-processing program should be able to incorporate the tables or graphs produced by another program performing an analysis of laboratory data. The ability to combine applications should also lead to interdisciplinary efforts. For example, programs that model thermodynamic systems could be used with a variety of discipline-specific applications.

Experience with the system has demonstrated that few applications in Athena's existing computing environment have coherent interfaces. Improvements in this area represent a major technical thrust of the project and should have a major impact on the use of computing at MIT.

New technology (i.e., bit-mapped displays and computers dedicated to a single user) and the potential for the comprehensive use of computing motivate the need to refine user interfaces. A "standard" framework for the user interface, common to all applications (including the operating system), can reduce the amount of training needed to use applications. New applications will emphasize interaction and graphics. If code for user interfaces can be generalized, it will reduce the amount of new code required to develop an application.

There are two ways of thinking about a standard framework for user interfaces. The first is to require that all applications use a single framework. The interface may be complex enough to require training, but because there is only one interface, training becomes a one-time investment. The alternative is to allow user interfaces to differ, but to require that all interfaces be self-explanatory.

The major risk of any standard framework for user interfaces is that a standard framework, if applied prematurely or blindly, suppresses innovation. Concepts employed in the design of a user interface span diverse areas of research (e.g., human factors and graphics), and there is little agreement on what contributes to a good user interface. Although Project Athena is addressing the issues of user interfaces, they

remain difficult issues, and we have modest expectations for success.

Ownership and operation. The campus network is owned and operated by MIT. Although the network's growth has been somewhat ad hoc, it has followed the general model of subnetworks linked by a "spine" network. The network has been subsidized by Project Athena (its largest client). A long-term plan for financing the network's operation and expansion is under discussion and will be based on operational experience with Project Athena.

Service computers are also owned and operated by MIT. As Athena's computational environment has stabilized, part of the operational support for its computers has been transferred to a centrally administered staff. The staff is responsible for providing computational support to a larger community consisting of the instructional research and administrative units of the university.

Project Athena has subsidized the acquisition of workstations. While public workstations have been provided by the project, private ownership can only be simulated by making workstations available to a subset of the campus community that uses them on an "experimental" basis.

In both cases, the user is expected to play an important role in operating the workstation. Users are responsible for managing the information they create (using removable media or network file services). They are also responsible for reporting failures and supporting repair activities (e.g., providing access to a private workstation). The existing computing infrastructure of MIT is now being expanded to include retail outlets for computers and facilities and services for repairing workstations.

Open issues

The three most critical and as yet unresolved issues that will have to be addressed by universities seeking to implement the proposed model of computation are information privacy, software licensing, and paying the costs of such a system.

The problem of maintaining the privacy of information on a network designed to encourage information sharing represents a formidable challenge. The characteristics of computer systems that make sharing easy are often precisely those that make protection of information difficult. Such problems can extend beyond protecting students and the university from the most obvious forms of abuse, such as widespread and largely undetectable cheating. As large distributed systems evolve, there will be a strong temptation for faculty and administrators to use the mail and other communications facilities to transmit potentially sensitive information such as student grades, staff evaluations, and salary-related data. Over time there will be strong reason to link MIT administrative machines into the Athena environment. This will present the opportunity of having a security audit by a group outside the academic computing environment. That is, the administrative side does not now connect its computers to the campus net and will not unless its data-security requirements can be met.

The resolution of these dilemmas will require a three-pronged approach. First, to the extent that is feasible, distributed systems must be made more secure without forfeiting the benefits of information sharing. This will probably require both the encryption of network traffic and secure ways of distributing keys. It is unlikely that these technologies will be cost-effective until encryption becomes a standard option on high-volume, low-cost network interfaces. Second, users will have to be made aware of the potential hazards of sending unencrypted messages on data between sessions, using easily decrypted passwords, and generally placing information—which simply should not be there—in the computer system. The third and most difficult task is to create a strong sense of ethics about information privacy within an academic community. It is evident from our experiences in Project Athena that there is no clear consensus about appropriate standards regarding the use of information within the university. Students who would not even consider invading someone else's paper files have less clear standards about looking at that same individual's electronic files. Universities may have to stress that "electronic" privacy has the same status as other forms of privacy, and they will have to work with students to make sure that members of the community have a common understanding of what constitutes appropriate behavior.

The problem of licensing software for large systems of networked workstations is an emerging concern for the software industry. Within Project Athena, we have struggled to develop site licenses that offer reasonable compensation to third-party software providers and that adequately protect their code, while still keeping costs reasonable and permitting use of the network for software distribution. For public workstations, strategies such as requiring "key disks" to run software or special hardware keys are almost impossible to manage. A more reasonable strategy is to have a network-based installation procedure that tailors the program so that

The Carnegie-Mellon Project

Activity similar to Project Athena is currently under way at Carnegie-Mellon University, a technically sophisticated institution located on a geographically compact campus. These attributes, in conjunction with a supportive administration, have given rise to a project to produce a campus-wide integrated personal computer environment.

Three interacting groups are working on CMU's computing future. The Computation Center has already deployed over 3000 personal computers for various purposes, the Information Technology Center is developing the software for a future system, and the Center for the Design of Educational Computing is facilitating the creation of educational software.

The Information Technology Center. ITC's computer system, named Andrew, after Andrew Carnegie and Andrew Mellon, is the prototype for Carnegie-Mellon's main campus computing system. It runs on the UNIX operating system. The two major components of Andrew are the communications system and the workstation operating environment. The ITC is sponsored by IBM, has been in operation since January of 1983, has expended about 70 person-years of effort, and has generated over 400,000 lines of program code.

The communications system. The model for the communications system is simply a file system like that found on a conventional time-sharing system. What is interesting is its size and internal structure: It is expected to have between 5000 and 10,000 active users, to hold millions of files, and to be implemented by a 10,000-plug high-speed network and hundreds of dedicated computers called file servers.

The first version of the network and file system was brought up at the end of 1984. In cooperation with the Computation Center and several academic departments, a campus-wide network of local area networks linked with fiber-optic cables and routing computers was created. Over 400 computers owned by various groups are connected to this network, all using the same communications protocol. Of these machines, about 100 run Andrew software and are supported by seven file servers. There are about 250 registered Andrew users. All users can send and receive electronic mail as part of the general campus mail network.

The workstation environment. Workstation efforts have been focused almost exclusively on the creation of general tools for exploiting a bit-map display and mouse. The building block for all applications is the window manager that virtualizes the display screen, dividing it into a number of rectangular areas whose size and shape are under control of the user. An extensive subroutine library provides the means to structure multifont, editable text inside a window.

The Computation Center. The CMU Computation Center has distributed over 3000 IBM PCs and Apple Macintoshes. At the end of 1984, the ITC began to deploy over 50 workstations to the university community, mostly for educational software development. This required a large set of activities beyond software development and buttressing the existing campus

Continued on page 123

it will run only on the machine to which it is downloaded.

Even with this type of arrangement, there is a question of what students may be allowed to take with them on graduation. If software is site-licensed to the university, does that license require graduating students to return copies to the university? Alternatively, does the license provide students with the right to take software with them, perhaps by exercising an option to buy at some predetermined fee? If alumni have access to the university-based software, will the university provide a mechanism for them to obtain updated copies as the software is revised? All parties want a resolution of these issues, but their uncertainty as to where the software market is going is great, and for many the risk of false steps is very high. At present the best that can be achieved is to maintain a good dialogue between producers, distributors, and consumers. This is clearly the role that MIT generally and Project Athena in particular can fulfill. To that end, Project Athena[4] hosted a roundtable at MIT's Endicott House in November 1984, and "it was clear that the parties representing the various academic, commercial, and government sectors needed to be brought together to gain a better understanding of the issues involved and the perspectives from which each came." (Proceedings published by MIT Industrial Liaison Program ; the preceding quote is from this report.)

The final issues are economic. Widespread use of computers is far more likely to increase the total cost of education than to decrease it—the payoff of com-

[4]See Commercialization of Educational Software, Project Athena, MIT, Cambridge Massachusetts, 1985, for a survey of the discussion. The conference was jointly sponsored by Project Athena and the Alfred P. Sloan Foundation.

continued from page 122

network: routine methods for registering users, user and programmer documentation, a hotline for system problems, and user consulting via electronic mail. Each member of the ITC technical staff served as a liaison to one of the groups using Andrew.

After an agreement with IBM allowing the distribution of the current version of the Andrew workstation software was secured, the software was publicized in UNIX circles and to the ICEC consortium.

The Center for the Design of Educational Computing. The purpose of CDEC, which is supported by a variety of sources, is to facilitate the creation of software to support undergraduate education. A few demonstration programs have been written to assess the potential of the Andrew system for graphical illustration of physical concepts: a game illustrating the properties of acceleration, an orbit tracing program, an optics workbench, a general mathematical function plotter, an automated aid to organizing layouts of information within windows, and an animation editor.

Implementation of the MicroTutor language, designed by Bruce and Judy Sherwood, is also well under way. This is the fifth version of the Tutor language begun at the University of Illinois in 1967. Over 10,000 hours of course material have been developed under various versions of Tutor, and while it is not likely that CMU will import much of that material, the existence of this proven tool will greatly enhance the development and importation of new software.

Finally, a large team from CDEC and the nascent Philosophy Department have embarked on a major effort in logic-teaching programs. The suite of logic software will be developed in both PC versions and Andrew versions.

Campus communications. Extensive time has been spent consulting with the campus community on all aspects of educational computing, from basic strategy to coping with programming bugs. Over three dozen campus educational software projects were funded, most targeted for Andrew. Software developers using Andrew meet weekly with various members of the ITC and CDEC staffs; there is a newsletter, a seminar series, and a catalog of educational software developed at CMU. The three dozen entries are limited to projects that have produced running software and to brief one- or two-page descriptions.

The CDEC Software Library has a full-time librarian, who is also on the FIPSE Technology Study Group, a conduit to educational computing and software database projects nationally. A database host will be chosen for the information being gathered, and collaborative opportunities with Michigan's Library Science School for research into classification and retrieval schemes for software are being explored.

Software publication. CMU has explored the possibilities for publication of educational software and will concentrate on a few high-profile, high-quality packages that run on IBM PCs and Apple Macintoshes—logic and calculus software. There is also interest in packages developed for music, operations research, civil engineering, and psychology, chemistry, and physics.

---

puters is improvement in the quality of education and not in reduced cost. Universities and their students will be faced with the need to pay for a major addition to the capital plant and a service staff to support the computational model we have described. It is quite possible that the total costs of the computing will exceed a university's expenditures for its libraries.

A logical division of the total cost would be for the university to treat the network and the back-end computers as a direct expense, with students paying for private workstations, either directly or through an identifiable increase in tuition. Financial-aid packages will also have to reflect this increased cost of an education, further straining what in most universities is already an overtaxed set of resources.

Computer centers will undoubtably move from their current status as vendors of CPU cycles to operators of a combination of a retail outlet (selling private workstations) and a utility (providing networking and specialized computational services delivered to workstations over that network). Some services may be treated as part of an operating overhead; others may operate on a fee-for-service basis. For example, network use could be treated as a part of the general university overhead, whereas printing charges might be based on usage. Some institutions may choose to adopt a modified version of the proposed computational environment. For example, it would be possible for a university to provide only public workstations and to charge students an appropriate usage fee. Although this approach would restrict the availability of computers and might result in contention for the limited set of private workstations, it would still allow for much higher utilization of workstations, thereby reducing the cost per user.

An often-discussed means of fi-

nancing some of the costs of a computation-intensive environment is the sale of university-produced educational software, but it seems unlikely that this will be a major revenue source. The market for educational software is small, and most of the institutional purchases are for generic software products rather than for subject-specific software. The total market is also highly fragmented, making it difficult for any one subject-specific software product to sell widely. In addition, university faculty further fragment the market by customizing their courses to reflect their own interests and the abilities of their students. Finally, mature distribution channels for such software are nonexistent.

Perhaps a more realistic source of financing is the university's alumni. Students may well find that certain network-based services such as library search, electronic mail, or archival service are of such value to them that they will be willing to pay for the services beyond their college years. In addition to providing a direct service to alumni, there would likely be enormous benefits for the alma mater in keeping alumni an active part of the extended community. New software could be made available to the alumni, providing a way for them to maintain their personal computing environment and at the same time reinforcing their ties to the university.

Even with such revenue sources, it is inevitable that some universities will lack the financial resources to sustain computation-intensive environments like those being proposed at MIT, Carnegie-Mellon, Brown, and various large, private universities. For these schools, the importance of this approach is not the extent of computing that it provides (e.g., a computer in every room); rather, it is the low cost of incremental addition. Our relatively heavy emphasis on creating a "single" system out of diverse and heterogeneous hardware ensures that the important part—the computer the student uses—can be added at little more than its unit cost. □

Edward Balkovich is a consulting engineer at Digital Equipment Corporation. As a staff member of the External Research Program, he is responsible for DEC's activities in Project Athena. His technical interests include distributed systems and operating systems.

Balkovich earned a BA in mathematics from the University of California, Berkeley, in 1968, and an MS and PhD in electrical engineering and computer science from the University of California, Santa Barbara, in 1971 and 1976, respectively. He is a member of both ACM and IEEE.

Steven Lerman is currently the director of Project Athena at the Massachusetts Institute of Technology, where he has been on the faculty since 1975. A professor of civil engineering, he has taught courses on computer algorithms, with specific applications to engineering and the sciences. He is the author of numerous technical articles and a coauthor of a forthcoming book on the statistical analysis of transportation demand.

Lerman received his bachelor's, master's, and PhD degrees from MIT in 1972, 1973, and 1975, respectively. His undergraduate degree is in civil engineering, and both graduate degrees are in the area of transportation systems.

Richard Parmelee is project manager for IBM's participation in Project Athena at MIT. Prior to that he was a staff member at the IBM Cambridge Scientific Center, where, among other things, he worked on virtual machine systems. His research interests also include file systems. Before joining IBM, he worked at MIT as a research associate. Parmelee holds a BS from the University of Illinois (1959) and an MS (1961) and PhD (1966) from MIT in mechanical engineering.

Questions about this article can be directed to the authors at Project Athena, Building E40, Massachusetts Institute of Technology, Cambridge, MA 02139.