Challenges of Utilizing Computer Technology In the Engineering Classroom

Keynote Address at the 1999 ASEE Southeastern Section Conference

James K. Nelson, Jr., Ph.D., P.E.[1]

Abstract

The personal computer is a tool that can be very effective in the process of educating engineers. Our challenge as educators is to provide students with the ability to recognize that the computer is a tool they can use both as students and as practicing engineers. Our students should also develop the ability to recognize that it is not the solution to either endeavor, and that the computer should not necessarily be used for the solution of all problems. In that regard, this paper addresses several questions regarding effective computer usage in the engineering classroom. These questions include the effectiveness of current utilization, what is effective utilization, and characteristics of good courseware. A very important issue discussed is the student in the classroom today and their expectations in regard to the computer. In addition, the evolution of the modern computer is traced.

Introduction

The focus this meeting of the Southeastern Section of the American Society for Engineering Education is the *Utilization of Computational Technology in the Engineering Classroom*. The technology about which we are primarily speaking, the personal computer, is just barely 20 years old. Interestingly, that is about the mean age of the undergraduates. This technology arrived on the heels of the electronic calculator and the profession has yet to determine how to effectively deal with calculators in the classroom, especially symbolic programmable calculators. The personal computer has created dilemmas that are exponentially greater.

The last six years has seen remarkable growth in personal computer technology and information available. When President Clinton was first elected in 1992, only a handful of scientists used the Internet. Although not strictly a true statement, it is about right. By the time he was reelected in 1996, the information superhighway became a 12-lane interstate flowing at capacity with minimal traffic control. The rapid growth of on-line connectivity began about 1995 when Microsoft® introduced Windows95®. That operating system directly (and conveniently) supported on-line connectivity. Now virtually everyone, especially students at our colleges and universities, have immediate access to a vast range of resources through their computer. This growth and the information now readily available have also caused problems for educators and at the same time provided them with unbounded of opportunities.

In addition to being a tool for collection of resource material, the personal computer with a windows-based operating system provides an excellent tool, if properly used, to visually demonstrate principles to students. It can provide a means for directed, self-paced learning outside the classroom. Our challenge as engineering educators is to effectively utilize the computer technology available in the process educating knowledgeable engineers. This is also our opportunity. In that regard, there are several questions that need to be answered. These questions include:

[1] Professor and Chair, Department of Civil Engineering, Clemson University, Lowry Hall, Box 340911, Clemson, SC 29634-0911

1. Are we being effective with computer usage today?
2. Where and how can the computer be effectively utilized?
3. What constitutes good educational software?

Through this paper, perhaps we can address some of these questions. The paper itself is a bit of personal philosophy on computer usage in the classroom, a bit of history, and the source of challenges regarding computer utilization. Hopefully this paper will foster creative thinking about how to use computers effectively in the engineering classroom. Recognize also, that as we finish this conference, we likely will have asked far more questions about effective computer utilization than we have answered. The problem we are addressing is very complex and there are no simple answers. It is a dilemma that has been brought about, almost exclusively, by the personal computer. As we begin our quest, it would be very interesting to understand the evolution of the computer. Part of the reason for this tour through history is to put into perspective some expectations of students today regarding computers.

Evolution of Digital Computer Technology

Computing is one of the oldest human activities. Charles Kettering, one of the most practical inventors of modern times, once said, "You cannot understand anything unless you can give it a number." This need for quantitative description has drawn civilization to develop computing devices (IBM, 1964). The first digital computer was the abacus which was devised in various forms by the Egyptians, the Greeks, the Romans, and the Chinese (Fishman, 1981).

Early Digital Computers

Development of the digital computer as we know it today began in the seventeenth century. In 1642 at the age of 20, a French mathematician named Blaise Pascal built the first operating digital computing machine. He devised a simple device for adding and subtracting which he hoped would be useful to his father, a government official who was revising the local tax system. Although the use of this machine was very limited, it did demonstrate the feasibility of mechanical computation. In recognition of his significant contribution to digital computing, the programming language PASCAL was named in honor of Blaise Pascal. At about the same time as Pascal, Leibniz conceived the idea of a computer that operated using binary code.

Despite the significant accomplishments of Pascal and Leibniz, the computer industry traces its origins to an Englishman named Charles Babbage who was one of the more interesting and innovative developers of machine computing (Brock, 1975). In 1882 Babbage devised the "Difference Engine" which was a small machine for computing polynomials. He applied to the Chancellor of the Exchequer and received funds to build a larger version of the engine. This was one of the first recorded government research grants. Babbage spent ten years on the second difference engine and was never able to complete it (Fishman, 1981).

In 1833, Babbage developed a machine on paper that could solve virtually any mathematical problem. He called this machine the "Analytical Engine." It was a mechanical device that operated with steam power and was able to perform controlled sequence calculations at the rate of sixty additions per minute (Fishman, 1981, Brock, 1975). Babbage's machine was similar in concept to modern computers. The plans included conditional memory transfer and the ability to remember up to 1,000 numbers with 50 digits each (Brock, 1975, IBM, 1964). Babbage worked on the analytical engine until his death in 1871, but he lacked the financial resources and the technology to build a successful model. We are aware of his work primarily through the writings of Countess Ada Lovelace, after whom the programming language ADA was named. Countess Lovelace wrote very descriptively of the design and the philosophy of the analytical engine. One of her statements about the engine stands as a preface to the study of modern digital computers. She wrote "The analytical engine has no pretensions whatever to originate anything; it can do whatever we know how to order it to perform" (IBM, 1964).

The Jacquard loom that had revolutionized the weaving industry in the early part of the nineteenth century inspired the analytical engine. The loom was programmed by punched cards; warp threads were pulled down through holes that were arranged to produce a predetermined pattern. In the analytical engine, punched cards controlled levers that moved barrels. Each barrel held an arrangement of studs representing a particular number or command. The studs were arranged in 80 columns like later computer punch cards (Fishman, 1981).

At about the same time as Babbage, George Boole, an English mathematician, was developing the algebra of logic. The algebra of logic is what we know today as Boolean Algebra. Boolean Algebra has become the foundation of machine logic, which is very necessary in the design of digital computers.

Another early inventor who deserves mention is Herman Hollerith. The U.S. Census Office hired him in 1879 at the age of nineteen. At the census office he was encouraged to develop a mechanized method to tabulate the census data. The tabulating system he invented used cards with holes to indicate sex, race, age, etc. The cards were passed under contact brushes that completed an electrical circuit whenever a hole was present. In 1886 Hollerith left the census office to form the Tabulating Machine Company which was subsequently sold to a primitive conglomerate that later became IBM (Fishman, 1981).

20th Century Digital Computer Development

Not much was done with digital computing machines until the 1930's when three men are generally credited with the development of modern digital computers. The first of these men was John Vincent Atanasoff who was born in 1903 and was the son of a Belgium immigrant. He received a degree in electrical engineering from Peddie and Colgate and a degree in theoretical physics from the University of Wisconsin. After that he taught at Iowa State University. While at Iowa State, he encountered the same problems that plagued mathematicians and physicists of that time. None of the available computational aids had the speed or the precision to solve the problems that were being studied. In 1935 he decided to resolve the problem and spent the next several years designing the digital computer. While in a tavern in the winter of 1937, four ideas crystallized for Atanasoff that made it possible for him to complete the computer. These ideas were binary code, non-ratcheting logic, serial calculation, and regenerative memory. He spent a year working out the details and applied to the Iowa State Research Council for an initial grant of $650 to hire a graduate student, Clifford Berry, and to buy materials to build an operating model. By the time the Atanasoff-Berry computer (ABC) was completed in 1939, Atanasoff had received two additional grants bringing the total to $1,460 (Fishman, 1981).

In December 1940, the second inventor of the modern digital computer visited with Atanasoff. John W. Mauchly was born in 1907, received a degree in physics from Johns Hopkins University, and became head of the physics department at Ursinus College near Philadelphia. By 1937 he realized that it would take a lifetime to perform the calculations necessary for his research in atmospheric electricity. In September 1941 he joined the faculty at Moore School of Engineering at the University of Pennsylvania where he met a young graduate student named Pres Eckert. Maulchy, along with Eckert, worked in the laboratory at combining Atanasoff's ideas with their own ideas. In April 1943, on Eckert's 28th birthday, the two men were funded by the U.S. Army Ordnance Department to build the electronic computer Maulchy envisioned. The computer was named ENIAC, the Electronic Numerical Integrator and Computer. The ENIAC project was completed in late 1945 at a cost of $486,805. Maulchy rejected Atanasoff's ideas on serial calculation and binary numbers which caused the machine to be more ungainly than was necessary (Fishman, 1981).

The third person in the development trio was the colorful Hungarian mathematician John Von Nuemann. He began to consult with the group at the Moore School of Engineering group as the ENIAC was being constructed. The designers began to discuss a second machine that might resolve some of the problems encountered in the first machine. The new machine was to be called the EDVAC, the Electronic Discrete Variable Calculator. In June 1945, Von Neumann wrote a 101 page paper entitled "First Draft of a Report on Edvac" which is considered to be a landmark paper in computing theory. It discussed the requirements for the design of a digital computer from a logic viewpoint. The paper also advocated binary code, serial calculation, and a method of operation resembling the human nervous system. These were the same ideas conceived by Atanasoff and built into the ABC computer of which Von Neumann had no knowledge. For the next three decades, all computers followed Von Neumann's model (Fishman, 1981).

In the late 1930's, IBM invested $500,000 at Harvard to develop the giant electromechanical device called the Mark I, which was originally named the Automatic Sequence Controlled Calculator. The machine was designed by Howard Aiken, built at IBM's Endicott plant under the supervision of three engineers provided by IBM, and was donated to Harvard in 1944. Aiken scarcely acknowledged the support of IBM in developing the computer, which was still a sensitive issue with the company 25 years later. The appearance of the all electronic ENIAC made the Mark I almost immediately obsolete. IBM responded by introducing a larger electromechanical computer called the Selective Sequence Electronic Calculator in 1948 (Fishman, 1981).

The next 30 years saw dramatic growth in electronic digital computing. Maulchy and Eckert formed a company that became the Univac Division of Remington Rand and introduced the UNIVVAC I computer in June 1951. IBM delivered its first electronic computer, the 701, in 1953 to compete with the UNIVAC. IBM later introduced the 650 in 1954. National Cash Register, Burroughs, and RCA developed other computers. All of these computers were designed around vacuum tubes (Brock, 1975).

The invention of the transistor by Bell Telephone in 1948 and its greatly reduced prices by mass marketing caused the second-generation computers to be built. Univac introduced the first transistorized computer, the Solid State 80, in August 1958. RCA was the next to deliver a solid state computer. IBM redesigned the 709, which was first delivered in 1958, with solid-state electronics and reintroduced it as the 7090 in 1959. IBM's popular scientific computer, the 1620, was introduced in 1959. Control Data Corporation was formed and introduced the scientific oriented 1604 in January, 1960. Digital Equipment introduced the PDP-1 in 1954 and Honeywell introduced the H-200.

The third generation for computers began around 1965 and was based upon integrated circuits originally developed by Texas Instruments. IBM announced the System 360 in 1964 and began deliveries in 1965. At the end of 1964, RCA announced the Spectra 70, which was designed to outperform the System 360. Control Data introduced the CDC 6600 computer in about 1965 and delivered the super-scale 7600 at the end of 1968. Since then there has been remarkable growth in the development of super-computers. These machines permitted engineers and scientists to solve problems that heretofore could only be conceived of in one's imagination.

Because of the digital computer, a "build and try" approach to engineering design was replaced by "mathematically model and simulate." The computer became a standard piece of design equipment. These major and far reaching accomplishments have all occurred in the past 40 years. Perhaps one of the most significant single events in engineering computation, however, was the introduction of the personal computer in the late 1970's.

Introduction of the Personal Computer

Personal computers were finally introduced around 1978. They were predominantly an outgrowth of the "electronic tinkerer." The original personal computers were purchased in kit form. As with the early mainframe computers, the early personal computers had very limited capabilities. They were not viewed as having much value for scientific computing by most people. The first models could not even multiply or divide; repetitive addition and subtraction was used to perform these operations. The first microcomputers in reality were expensive toys whose real value to the owners was the challenge to make them run. Virtually all programming of these computers was done in machine language. Soon a BASIC language interpreter was developed. This command interpreter was very slow but it did significantly reduce the programming effort. With this improvement, applications were found for the early microcomputers. They were advertised as "home computers" for checkbook records, address directories, and recipe files. There was not much use for these applications in scientific computing.

Fortunately, however, a few people did recognize the impact that personal computers could have. Engineering computing was attempted with fair success in the late 1970's, actually about 1979. At the ASCE Seventh Conference on Electronic Computation in August, 1979, Dr. Milo Ketchum (1979) presented a paper entitled "Structural Analysis for Home Computers." His programs were capable of analyzing small to medium sized frame structures, which was a significant accomplishment in its day. In hindsight, there was something even more significant about his presentation. At the conclusion, Dr. Ketchum gave copies of these programs, at no charge, to anyone who wanted one. The software market has changed drastically since the early days of personal computers.

For the next couple years, people were looking for scientific applications for personal computers and the computers themselves began to grow in size. There is a paper of historic interest that was presented in May 1981 at the First International Conference on Computing in Civil Engineering. This paper by John Thigpen (1981) was entitled "Microcomputers in Small and Medium-Sized Firms." In this paper, Mr. Thigpen wrote:

> *Up until recently, the microcomputer has exhibited a trend of lowering costs. This trend is gradually reversing, and all indications are that the bottom on cost has been reached. Rather than continued lowering prices, movement is seen in the direction of increased sophistication, capabilities, and increasing costs. Several technological advances are foreseen. The pending development of the 16k RAM (Random Access Memory) chip will soon allow for the doubling of internal memory of some microcomputers.* (Thigpen, 1981).

As it turned out, these predictions were rather shortsighted. First, the cost of personal computers continued to decrease after seeing only a modest increase. Today, the cost is lower than it has ever been and the direction it will go in the future is unclear. Second, technology advanced much quicker than could have been imagined. Two years later, personal computers had 256k RAM, sixteen times that which was pending in 1981. By 1986, that capacity had more than doubled.

The period from 1979 to 1983 was a turbulent time for the personal computer industry. "Chaotic time" may be a better description. Many manufacturers produced computers, each with their own operating system. The different machines could not communicate with each other. Disks created on one machine could not be read on any other machine. This situation was stabilized, however, around 1983 when IBM introduced the PC. With the PC, IBM set the standard for the personal computer industry. Quickly, "IBM compatible" computers appeared on the market. Software and data could finally be transferred from one computer system or user to another. Technical software could finally be developed and sold to a much wider market.

During the next few years, much software was developed and marketed, and the personal computer began to entrench itself in the engineering office. Each engineer began to have his or her own computer. Some of the personal computers had more capability than the mainframe computers of thirty years ago. Some schools now require entering engineering freshman to acquire a microcomputer. In 1979, Professor Ed Wilson (1979) presented a paper entitled "Role of Small Computer Systems in Structural Engineering" at the Seventh Conference on Electric Computation. In this paper, he predicted that by 1990 engineers would be using personal computers exclusively and that these computers would serve as terminals to other computers. Wilson's predictions, which were made at the beginning of the microcomputer revolution, were quite accurate.

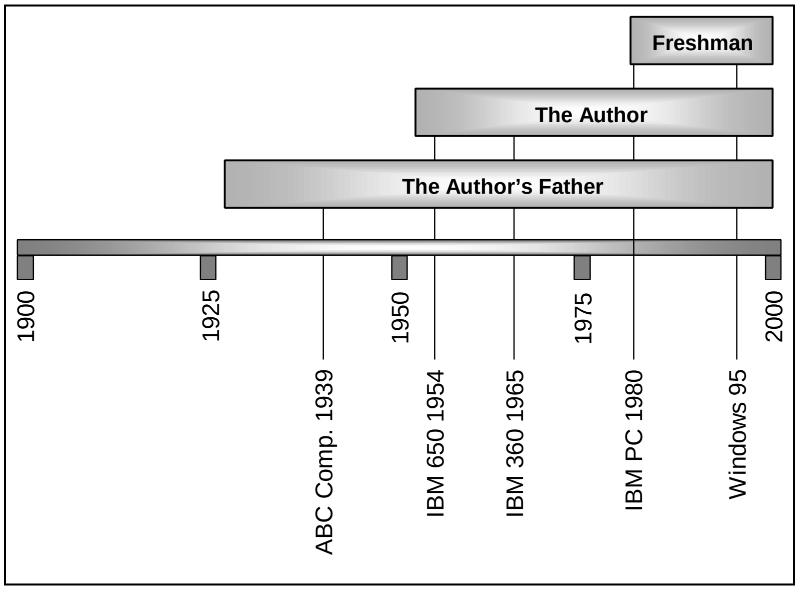

Figure 1– Timeline of Modern Computers and the User

Effective Computer Utilization

The personal computers used in education today have windows-based operating systems that are themselves graphically based; the software that we deal with is almost all graphically based. Indeed any software operating in a windows environment is graphically based. This graphical basis has lead to presentations that can be rich in visual content and documents that are pleasingly formatted. These are things we could not easily do before the personal computer became so available. These are things that we have come to expect as educators. It also has led to the development of phenomenal interactive games that our students have played as they grew up.

As we talk about effective computer utilization, we must talk about the students that are using the computer as part of their education. To understand the undergraduate student studying engineering today we must recognize the period in which they grew up as opposed to the period in which we grew up. Consider the illustration shown in Figure 1. Also consider for a moment the computational environment in which the freshmen this fall grew up. Compare that to the computational environment in which you and I, and our parents, grew up. We had computers during our high school and college years, but these were large machines that used punch cards or a remote terminal to receive instructions. We did not expect to be actively involved with the computer. Looking further at the timeline, we were already practicing engineers when the personal computer was introduced.

On the other hand, the freshmen next fall had personal computers, with the wealth of interactive graphics they offer, for their entire life. Windows 95® and dramatic growth of the Internet occurred the year they started high school. The personal computer is a tool with which they are very accustomed. They expect to be involved with the computer and they expect the presentation of information to be highly graphical, to be very visual.

The E-Generation

As we begin to address effective utilization of computers in the classroom, we need first to look at the students we are teaching. We are attempting to teach students who have wholly grown up in the electronic age. These young people have been called e-kids and the e-generation. On MSNBC "The Site" (MSNBC, 1997) today's students were referred to as cyber-children– children who have been raised completely in an age of interactive graphics, in an age of interactive technology. The teaching methods used to educate these students must change to accommodate the environment in which they were raised. The methods by which we learned are not necessarily the methods that will enable students today to learn. The question, then, is how can students be motivated to achieve knowledge and comprehension of engineering subject matter and be able to apply and analyze that knowledge (Nelson, 1998). Stated differently, the question is: "How do we take them from where they are now to where we need them to be?"

Two developments have occurred in recent years that dramatically impact the means through which students learn– the means through which they are transformed into knowledgeable engineers. The first of these is educational technology, computational technology, with its rich capability for visual representation (Kaha, 1990). Today's students have grown in a society that depends heavily on television and multimedia for information. They have learned to learn by visualization, with less emphasis on learning by listening. To a great extent, though, the visualizations used are visualizations prepared by others– the student is not an active participant in the process (Nelson, 1998).

The second development that should affect the manner in which students are taught is a better understanding of how learning occurs. In the past 20 years cognitive psychologists have advanced significantly the understanding of how students perceive, process, store and retrieve information (Embretson, 1995; Jarman and Krywaniuk, 1978; Waldrop, 1987). These understandings have led to a new learning paradigm where the instructor's emphasis should be on the learner's capacity to effectively process information presented through different sense modalities. This change should lead to dramatic alterations of the instructor's role as was previously suggested (Nelson, 1998). Emphasis on the capacity to process information should also affect the manner in which information is presented to the students and the exercises they are expected to perform.

How Can the Computer be Employed

There are numerous areas in the educational process in which the computer can be effectively used. These areas can be generalized into three broad categories, namely: dissemination of information, gathering of information, and development of knowledge. Of these uses, the first two are rather pedestrian and are included here mostly for completeness. As educators, the latter usage, the development of knowledge, is the topic upon which I think we should focus most of our efforts. Nevertheless, let us talk about each of these areas for a moment.

The first use was for the dissemination of information, primarily the dissemination of course material. Much of the information provided to students, in fact virtually all of the information provided to students, can be disseminated electronically through the Internet. This information includes course notes, homework solutions, and sample data for problem testing. By placing the information on the Internet, the responsibility for obtaining the information rests with the students. The computer can also be used as the means of presenting lectures. Neither of these uses are particularly novel– both have occurred for several years.

The computer also can be used as a resource gathering tool. Information can be obtained through on-line databases and through a search of the Internet. This utilization is something that students have been doing for a while and with which they are quite comfortable– some parents think their children are spending entirely too much time browsing the Internet. There is a down side to this usage, however. Because of the ease with which on-line searches can be conducted, students are beginning to view such capability as the only manner in which to search for information. Much information needed by students in their studies is not on the Internet, nor is it likely to be any time soon. This is particularly true of legacy journals, conference proceedings, and textbooks. Students need to understand that the computer is a tool in the search for information and that it is not the only tool.

The last utilization, development of knowledge, is where we will spend most of our time in this discussion. This point is the thrust of effective computer utilization in the classroom– utilization that directly benefits the educational process in a manner that could not otherwise be accomplished, or in a manner that alleviates the need for repetitive calculations. Visual and interactive software that is contributing to the learning process– contributing to the understanding process by causing the student to think and act– is effective utilization of the computer in the classroom. This is the utilization on which we want to focus much of our discussion. It can be embodied in virtual experiments that the student must conduct, in tutorial systems that demonstrate principles and provide problems for the student to solve, and in a graphical and animated demonstrations used as part of an instructor's lecture in class.

What is effective computer utilization

A modification occurring in the engineering classroom is the incorporation of computer-aided learning with extensive graphics and animation. Specifically, the change is the type of software that is available and the way that software is used in the classroom. The focus on the use of computers in the classroom– or more broadly, in the learning process– must be directed toward enhancing the students learning and comprehension of the subject matter through the visual modality. The focus should not be on enabling the students to solve larger problems more quickly (Nelson, 1998). Nor should the focus be on doing with the computer what was traditionally done with the chalkboard and the copying machine.

Effective computer utilization in the engineering classroom is any utilization that causes the student to think, to extend or reinforce his or her current knowledge. To be effective, the software used, or the tutorials developed, must utilize the learning modalities with which the student is accustomed– the visual and aural modalities. Further, the demonstrations and tutorials must be interactive. The student must be a part of the system if it is to be effective. Although learning styles have changed, the benefit of "learning by doing" has not changed. A student actively involved in the process is likely to learn more than is a student who is a passive observer.

While talking about effective utilization, we should speak for a moment about what is not effective utilization. In particular, effective utilization of computational technology is not using the computer as the solution to the problem. Engineering students should not be taught to solve engineering problems by entering data into a purpose-specific software package. Often we lapse into this mode only because we find such a quick solution exciting. But we know the principles. We understand the solution and recognize that the program being demonstrated is nothing more that an implementation of the solution principles. In my opinion students *must absolutely be taught the principles of the solution*. They then can be shown "black box" software as a method to implement the solution, or as method to better understand the implications of the solution by quickly conducting parametric studies to develop an understanding of system behavior.

One needs to be a little careful in applying the thoughts of the previous paragraph. The purpose of a course must always be considered when determining effective utilization. In a senior capstone course, use of canned analysis and design software is probably very appropriate. The emphasis of such courses is usually to bring different disciplines together to enact a solution, much as is done in industry. The principles should already have been learned. Likewise, using an analysis package in a design course is probably appropriate. However, using design software in that course probably is not appropriate. Again, the guiding principle is to determine the best method to present and reinforce the principles that are to be taught in the course.

Programming vs. Computational Tools

When talking about utilization of computers in the classroom, the discussion invariably turns to a discussion of whether programming should be taught as part of the undergraduate engineering curriculum. Twenty-five years ago the only way to avoid a hand solution to a problem was to program the solution; this situation by itself was probably sufficient reason to learn programming. Today, however, there is a plethora of computer-based computational tools available that do not require programming. With the availability of these tools, does programming need to be taught? Since the advent of the personal computer this debate has become very heated and the two camps do not seem to be converging.

Personally, I have completely changed positions on this issue over the past few years. I was a strong advocate for programming in the engineering curriculum and in 1993 co-authored a paper (Nelson, 1993) that in part advocated keeping programming in the engineering curriculum. Since then, I have become quite convinced that a proper engineering education does not require instruction about programming. However, interested students must have the opportunity to learn engineering programming as part of their program of study– students likely to go on to graduate school might even be counseled to take a programming course. The operative phrase in the last sentence is *engineering programming*. Engineering programming is not necessarily programming as taught by computer scientists. The focus of engineering programming is developing software that solves a particular class of engineering problems– the emphasis is on obtaining a reliable and reusable solution to a complex problem. Let us explore the issue of teaching programming as we discuss effective computer utilization in the engineering classroom.

There is the age-old statement that if you can program the solution you understand the solution. This statement probably has some value; I believed it myself for many years. In hindsight, however, I do not believe it to be a true statement. Even when programming, students can muddle through a solution and still not understand what they are doing. Their programs look like their hand calculations. The only difference is that logic jumps replace the lines and arrows telling us where to go next.

The likely genesis of the statement that programming a solution demonstrates understanding of the solution stems from the common belief that programming requires an orderly step-by-step solution procedure. When one can identify all of the steps in the solution process, order those steps in a rational manner, and associate the appropriate computational algorithm with each step one likely understands the solution. Programming, in principle, causes this to occur because these are the steps necessary for developing a program. This is not how most students approach programming, though. They simply jump in and begin to write code in the same manner they jump into hand calculations. On the other hand, if a student is required to outline the steps in a solution process in detail (essentially prepare the flowchart the program) and then demonstrate the solution with an example, have not we as educators really achieved our purpose? Today, I believe that we have accomplished our purpose.

A related issue is what constitutes programming. Does programming only include development of code in traditional languages such as C or FORTRAN, or could programming include developing solutions using a computational package such as MATLAB® or MAPLE®, among others. I contend that it does. Software packages such as these are tailored to engineering applications and provide the ability to develop and integrate logic into a solution. They are, in essence, very high level interpretative languages. Solutions using these packages involve all of the elements of traditional programming including logic necessary for any iterative solution and a prescribed syntax. The advantage of using these packages over a traditional programming is that they contain many primitives (*e.g.* equation solvers and plotting routines) that permit the student to concentrate more on the solution than the method. The result of such an endeavor is essentially a program; when teaching with these packages we are essentially teaching programming. Solutions submitted using these packages, however, are as difficult to read as solutions in FORTRAN.

I believe that our true goal as engineering educators is to enable students to solve engineering problems and understand the behavior of engineering systems. Keller (1987) suggested that the computer be used "as an intelligent pencil." To this end, our students may be better served by providing them with instruction about a calculation package such as MathCad®. Packages such as these enable students (engineers) to perform calculations in a manner similar to hand calculations, to easily change problem parameters, to intuitively present information graphically, and to incorporate logic for iterative solutions. The results look like engineering calculations, the methods used are more apparent, and the result can be reviewed as engineering calculations. Brannan and Murden (1997) in this forum have discussed such use of software in the classroom. The student presents the methods of the solution but does not need to perform the mathematical operations. In a given period of time, this enables the student to spend more time developing an understanding of the solution and system behavior, and less time performing mindless mathematical operations. Further, calculation packages can be effectively used in all of a student's engineering classes. Students have gained knowledge about a tool that will benefit their entire education. This is simply not the case with traditional programming or with computational packages. This is the reason that I strongly advocate incorporation of calculation packages into the curriculum rather that programming. In this manner the computer has become an electronic pencil, an extension of the person.

Effective Didactic Software

The prior discussion leads to the issue of what software is appropriate and how it should be used. There is not a single response to this query because what is appropriate in one course may not be appropriate in another course. As already stated, commercial analysis and design packages may be appropriate in a capstone design course, but most faculty will agree that such software is likely not appropriate in an introductory analysis course. The fundamental question to be asked is "Does the software enable the student to obtain a better understanding of the course content using his or her preferred learning style?" Content in the context of this discussion is the set of fundamental principles that are to be learned and applied through study in the course. The ability to successfully execute software and generate large quantities of data is not necessarily the acquisition of knowledge.

Keller (1987) stated that:

> *"Only when we exploit the interactive nature of computers can we see what makes courseware different from other instruction. That is not to say, of course, that a graphic cannot be there just to delight. But it does mean that the computer, unlike any other media except the live teacher, allows– indeed demands– interaction. ... (There is) nothing wrong with an attractive display; but a graphic in a lesson should be there to solve a very particular instructional problem for which we think a visual image is the best solution."*

Keller went on to provide guidelines for development of graphical courseware. These guidelines are:

1. A visual image renders concrete and accessible what may otherwise be abstract and difficult.

2. Courseware graphics must be more than static images of books or even animated images of film: they must be interactive.
3. Using the computer as an intelligent pencil provides students with visual analogues of their thinking. The computer substitutes for their hands, not their minds.

This latter point is perhaps the most important point in the development of courseware for engineering education, and indeed the entire utilization of computers in education. It is a point to which we made reference earlier. The computer should relieve the student of the need to perform tedious and repetitive calculations. It should not relieve the student of the need to think.

Proper software for use in the classroom likely will require development. Hopefully this software would then be universally distributed through the Internet to avoid duplicate development efforts. Despite development efforts in the past, there is not much instructional software available for students working on personal computers. Effective software for use in the classroom today will have several significant attributes. These include:

- modules directed at specific content with sufficient problems and problem variations to provide virtually unlimited drill for the students,
- a graphical interactive user interface that can operate effectively on the class of personal computer owned by the typical student,
- an animated demonstration and interpretation of engineering system response,
- the ability to provide immediate feed back to the student regarding progress,
- the ability to interpret mathematical expressions typed by the student and to evaluate the correctness of the solution represented by those expressions,
- the ability to recognize a correct but alternate solution to the problem, and
- the ability to assist a student when "all hope is lost."

Development of course software incorporating these attributes is not an insignificant development effort. It does provide two benefits, though, for students when they are taking the course and in later courses for which the information is a prerequisite. The first benefit is faculty independent, supplemental, self-directed practice of the principles developed in class. This will enable the faculty to spend more time with the principles of the solution and less time repetitively working problems for the student. Through the intended interactive nature of the software the student has become an active and integral part of the process. Secondly, the software will provide self-directed remedial work for students in later classes when the early principles have become "dusty."

Who Develops Didactic Software

Who should develop software for engineering education is one of the most difficult questions to answer. In principle, it is not a difficult question to answer– engineering faculty in collaboration with faculty from education and computer science should develop engineering educational software. What makes the question difficult to answer is the environment in which faculty must survive. Faculty must quickly develop an externally funded research program in their area of expertise and publish the results of that research in peer-reviewed journals to survive the tenure and promotion process. Further, the results of that research must be viewed as having made a significant contribution to engineering knowledge. In the current environment a department chair or dean is not likely to recommend, much less urge, an untenured member of the faculty to pursue software for engineering education as their contribution to the institution. Developing software for education is outside the norm of engineering research, and we all know that anything outside the norm is very difficult package for promotion and tenure.

So then, who is going to develop software for engineering education? I still contend that engineering faculty collaborating with faculty from education and computer science must develop such software. I further contend that developing such software is not inconsistent with expectations of the current environment if one examines the intent of those expectations. Those expectations could probably be simply stated as follows:

1. Faculty are expected to develop an externally funded research program.
2. Faculty are expected to disseminate the results of that research.
3. Faculty are expected to make a significant contribution to the profession.

Let us examine these generalized expectations and see if they are not consistent with the development of software for engineering education.

First, developing software that achieves the goals outlined earlier in this paper is not an insignificant task. There is considerable research involved in defining the problem solving process and in developing algorithms to permit open-ended solutions. Projects such as these are fundable by the National Science Foundation under its undergraduate program initiative, and other agencies. As such, it is possible to develop an externally funded research program in educational software development. The individual needs to have talent in this area and to aggressively pursue funding. If faculty from education and computer science are involved, the development process become a multi-disciplinary activity which is strongly encouraged today.

Secondly, if the software developed is beneficial and is made available it will be adopted and used by other educators. This is dissemination and utilization of university research. In fact it goes back to the root of a university's primary mission of education. Acceptance and utilization of the software by other institutions should be considered equivalent to the peer-review process. Publication of the methods and innovative procedures in development of the software itself can be published in peer-reviewed journals. As such, the intention of the requirement for publication and dissemination of research results can be accomplished.

Third, is it not a significant contribution to the profession to improve the educational process and thereby improve the knowledge of our graduates? Such endeavors offer benefit to the students and are thereby an indirect benefit to the engineering profession. The profession can only improve through a workforce that is better educated.

Of these three points, the last two are the most difficult to quantify for promotion and tenure evaluations. They both require much more understanding on the part of the evaluators than does counting papers in particular journals and the number of times those papers have been referenced by others. It also requires an understanding of the effort and creative thought necessary to develop software that is accepted and used. Few faculty have an understanding of the effort and that understanding can only be developed through involvement in software development. This creates a bit of a "chicken and egg" situation. How can people developing educational software be fairly evaluated until people who have developed educational software have come through the system? It requires much insight on the part of those presenting an individual's qualifications and a willingness to take a position regarding the quality of work that outside the norm and not necessarily popular.

Conclusion

Without question, the personal computer has become an extremely effective engineering tool. If used properly, it can be an extremely effective tool in engineering education. As we have discussed at length, proper usage must focus on the educational process. Keller indicated that the computer should become the student's pencil rather than replace the student's brain. Stated differently, the computer should become a tool used in the search for an understanding of engineering principles; the computer should never become the solution itself.

I firmly believe that each and every one of us wants to effectively use the computer in the process of educating our students. I also believe that as a profession we have not gotten to that point. There are pockets of good software and effective utilization, but those pockets are not universal. To achieve effective utilization of computational technology in the classroom, and to continue improving on that effectiveness once it is achieved, I offer you the following challenges as we close this discussion.

Challenge One

The first challenge is one of self-evaluation. Each of us needs to look at the courses we teach and ask three questions. These questions are: "Can technology be effectively incorporated into this course," "What type of computational technology is appropriate for this course," and "Is the computational technology being use adding to the educational process or is it becoming the solution." These are very important questions and must be answered honestly before effective utilization can be achieved.

Challenge Two

The second challenge is a philosophical challenge that involves a definition statement of what is effective utilization of computational technology. This august body should develop a statement on computer utilization in the engineering classroom and laboratory. That statement should define in philosophical terms what is intended by effective utilization in the engineering classroom. It should go on to clearly prescribe characteristics of effective software and how that software can be evaluated. Such a statement will clearly set the tone for software used in engineering education. The statement should also include statements about what classes of software are appropriate for different educational levels. It is a statement that would provide significant guidance for engineering programs.

Challenge Three

The third challenge involves the educational institutions. Engineering departments and colleges should adopt a statement regarding effective utilization of computational technology of their own. This policy should be based on the statement prepared in the second challenge, but should be extended for the particular institution. One of the ABET EC2000 criteria is utilization of computers in the educational program. Adoption of such a policy demonstrates the importance a program places on effective utilization of computational technology. It also provides a clear map for satisfactorily satisfying that criterion.

Challenge Four

The fourth challenge involves you and I, and software for education. We should begin development of software for the classroom and to provide for the dissemination of that software. Someone has to begin the process and it may as well be us. The software should be in accordance with the guidelines discussed above and should be consistent with the policy statement developed by this body. In this process we should utilize the talented people in education and computer science, *i.e.* we should actively work with those people in the development process. The faculty also should incorporate other effective visual computational means for presentation into their courses to enhance student understanding.

Challenge Five

The last challenge is directed to department chairs, deans and provosts at our academic institutions. It involves putting a reward structure into place for members of the faculty who are developing educational tools. Each individual should develop an understanding of the rigor necessary to develop educational software. Each should incorporate into their respective promotion and tenure guidelines an explicit statement regarding development of educational tools and criteria by which the faculty developing the tools can be evaluated.

References

Brannan, K. P. and J. A. Murden (1997) "MathCad as an Alternative to a Traditional Computer Programming Language," *Proceedings of the 1997 ASEE Southeastern Section Meeting*, American Society for Engineering Education, Marietta, GA.

Brock, G. W. (1975) *The U.S. Computer Industry*, Ballinger Publishing Company, Cambridge, Massachusetts.

Fishman, K. D. (1981) *The Computer Establishment*, Harper and Row, New York.

IBM, *An Introduction to Engineering Analysis for Computers*, White Plains, New York, 1964.

Kaha, C. W. (1990) "Learning Environments for the Twenty-First Century. *Educational Horizons*, Fall Ed., pp. 45-49.

Ketchum, M. S., "Structural Analysis Programs for Home Computers," *Proceedings of Seventh Conference on Electronic Computation*, ASCE, St. Louis, August 1979.

Keller, Arnold (1987) *When Machines Teach: Designing Effective Courseware*, Harper & Row Publishers, New York.

MSNBC (1997) "The Site," Comments made by invited guest and commentator on 2 July 1997.

Nelson, James K., David H. Reilly, and Russell H. Brown (1998) "Teaching Engineering to Cyber Children," *Proceedings of the 1998 ASEE Southeastern Section Meeting*, American Society for Engineering Education, Orlando, FL.

Nelson, James K., Dennis J. Fallon, and Russell H. Brown (1996) "Re-Engineering the Civil Engineering Classroom," *Proceedings of the 1996 ASEE Southeastern Section Meeting*, American Society for Engineering Education, pp. 23-28.

Nelson, J. K., and John Alden Murden (1993) "A Case for Using Computers in Structural Engineering Education," ASCE *CE Computing Review*, Volume 5, Number 3.

Thigpen, J. E., "Microcomputers in Small and Medium-Sized Firms," *Proceedings of First International Conference on Computing in Civil Engineering*, ASCE, New York, May 1981.

Wilson, E. L., "Role of Small Computer Systems in Structural Engineering," *Proceedings of Seventh Conference on Electronic Computation*, ASCE, St. Louis, August 1979.

.

James K. Nelson, Jr., Ph.D., P.E.

Dr. Nelson graduated from the University of Dayton in 1974 with the degree Bachelor of Civil Engineering. He received a Master of Science degree and a Ph.D. from the University of Houston in 1976 and 1983, respectively. He taught at Texas A&M University for ten years before joining the Civil Engineering Faculty at Clemson University in 1989. Currently, Dr. Nelson is a Professor of Civil Engineering and Chair of the Civil Engineering Department. Previously he served as Program Director of the Clemson University Graduate Engineering Programs at The Citadel in Charleston, SC. Dr. Nelson is a Fellow in the American Society of Civil Engineers and a Member in the Institute of Marine Engineers. He is a registered Professional Engineer in four states and a Chartered Engineer in the United Kingdom.

Since joining Clemson University, Dr. Nelson has taught graduate and undergraduate courses in structural engineering, numerical methods, and programming for engineering applications. Many of the graduate courses have been taught on television as part of the distance education program at Clemson University.

Dr. Nelson has co-authored an undergraduate structural analysis textbook and has developed software for three structural analysis and design textbooks. In addition, he has developed much design and modeling software for industry. In the Civil Engineering Department, Dr. Nelson was one of the pioneers in dissemination of course material in distance education programs via the Internet.

Prior to receiving his Ph.D., Dr. Nelson worked as a design engineer in industry where he was primarily involved in design of floating and fixed structures for the offshore petroleum industry. Dr. Nelson's primary research interests are in the behavior of structural systems. For the past several years he has been actively involved in evaluating the behavior of free-fall lifeboats and the development of analytical tools to predict that behavior. He has authored many technical papers that have been presented in national and international forums. Dr. Nelson has served as a technical advisor to the United States Delegation to the International Maritime Organization and in that capacity is a primary author of the international recommendation for testing free-fall lifeboats and many of the international regulations regarding the launch of free-fall lifeboats.