Assigned: 31 Jan '07 Topic: What is engineering? (part II)

Required:

Petroski, Henry (2003), "Early [Engineering] Education", American Scientist 91 (May-June): 206-209.

Loui, Michael C. (1987), "Computer Science Is an Engineering Discipline" Engineering Education 78(3): 175-178.

Strongly recommended:

* Re-read (or finish reading) Brooks 1996

ENGINEERING

Early Education

Henry Petroski

Children are born engineers. Everything they see, they want to change. They want to remake their world. They want to roll over, crawl, walk. They want to make words out of sounds. They want to amplify and broadcast their voice. They want to rearrange their clothes. They want to hold their air, their water, their fire, their earth. They want to swim and fly. They want their food, and they want to play with it too. They want to move dirt and pile sand. They want to build dams and make lakes. They want to launch ships of sticks. They want to stack blocks and cans and boxes. They want to build towers and bridges. They want to move cars and trucks over roads of their own design. They want to walk and ride on wheels. They want to draw and paint and write. They want to command armies and direct dolls. They want to make pictures out of pixels. They want to play games—sometimes computer games. They want to talk across distance and time. They want to control the universe. They want to make something of themselves.

click for full image and caption

Grown-up engineering, which is as old as civilization, maintains the youth, vigor and imagination of a child. This is why, when presented to children on their own terms, the excitement of engineering is immediately apparent and fully comprehendible. No child is too young to play and therefore to engage in engineering, albeit of a primitive kind. We all did so as children ourselves, when we devised our own toys and games—and sometimes even imaginary friends to enjoy them with us. The idea of playfulness is embedded in engineering through the concepts of invention and design. Not that engineering is frivolous; rather, the heart of the activity is giving the imagination its head, reining it in only to check impossible or dangerous dreams and to turn ideas into reality.

Children do experience the essence of engineering in their earliest activities, yet there is seldom any recognition that this is the case. They may hear the word "engineer" only in connection with railroad locomotives and have no idea that their playful activity could become a lifelong profession. Engineers themselves are understandably reluctant to equate their professional activity with mere child's play. After all, they studied long and hard to master esoteric knowledge of atoms and molecules, stresses and strains, heat and power, currents and voltages, bits and bytes. They manipulate equations, not blocks. They use computers for serious modeling and calculation, not for fun and games. They design and build real towers and bridges that test the limits of reliability and safety, not toy ones that totter and fall down with little consequence.

These regimens learned in college and put into practice are important and serious, but they are still not essential to comprehending the profession's fundamental activity: design. Design is rooted in choice and imagination—and play. Thus the essential idea of engineering can readily be explained to and understood by children.

Sharing the Joy

Much has been said and written about the declining numbers of and disappointing lack of diversity among college students majoring in engineering. Among the factors cited to explain this paucity are the lack of exposure of high school students to the very idea of engineering and the fact that many have insufficient mathematics and science background to gain entrance to engineering school, even if they do identify the profession as a possible career. This is unfortunate, for the ideas of engineering should be integrated into the curricula not only of high schools but also of middle and primary schools. Our children are being done a disservice by not being exposed properly throughout their education to engineering activities identified as such. After all, even preschool children have the prerequisites in their play for appreciating exactly what engineering is: design. Indeed, design is ubiquitous throughout their school day, even in their before- and after-school activities. It need only be pointed out to them that they are designing something, and therefore being engineers of sorts, in virtually everything that they do.

According to Nicholson Baker in his novel, *The Mezzanine*, "Shoes are the first adult machines we are given to master." We learn to tie our shoes even before going to school. This is no mean accomplishment, as most of us may remember, and its execution is by no means as rigidly codified as the alphabet drilled in school. There are different ways to tie a shoelace, as we readily learn when we help different children unknot theirs, and the steps in tying a knot or bow can vary from family to family in ways that the order of the letters in the alphabet cannot. Most children learn from their parents, and in their teaching role the parents themselves often have to relearn from a different point of view. That different techniques exist is characteristic of the fact that tying a shoe is a design problem—and design problems seldom if ever have unique solutions. Each child may be taught to tie shoes in a prescribed way, but it is not the only way or even necessarily the best way. Such an observation is beneficial not only for introducing students to design but also for augmenting lessons in diversity.

Opportunities in the Everyday

The idea of tying a shoe, and the related problem of lacing one up, can be turned into playful educational activities that expose students to the idea of design and thereby to engineering. A recent article in the *New York Times*' "Science Times" described how Burkard Polster, a mathematician at Monash University, calculated that there are more than 40,000 distinct ways to lace up a shoe with two rows of six eyelets each. In true academic mathematical fashion, Polster extended his research by viewing the laced shoe as a pulley system to determine which lacing pattern was most effective in performing its function. He also determined the lacing that could be effected with the shortest lace. The combinatorial mathematics used by Polster make the problem as he approached it unsuitable for young children, of course, but the practical problem itself can definitely be used to advantage in the elementary school classroom. How much fun could children have redesigning the lacings of their shoes into imaginative patterns and learning by doing that there is more than one way to solve a problem? Being told by the teacher that a mathematician calculated that there are exactly 43,200 different ways they could have solved the problem can only add to the wonder of the lesson.

Elementary school students might also be asked if they could imagine how Polster got the idea of counting how many ways there are to lace a shoe. Telling them that he did so after learning that two physicists from the University of Cambridge calculated how many ways there were to knot a necktie provides yet another opportunity to to describe a commonplace problem in design. Even if the children are not in uniform—and the teacher, too, is dressed casually—the tie-knotting problem is at least one they might take home to tackle with their families. It would also expand the vocabulary of professions to which the children are exposed. To their knowledge that mathematicians can have fun counting shoe lacing patterns, the students can add the mental note that physicists can have fun counting tie knottings. To this can be added the observation that if mathematicians and physicists have such fun counting things, imagine how much fun engineers have in designing things that can be counted.

(As an aside to teachers and others, the word "science" is in fact a misnomer when it actually refers to engineering. Science, strictly speaking, does not include engineering, an activity distinguished by its domination by design. Engineers design things, such as patterns of shoe lacings; scientists analyze things, such as counting how many lacings can be designed. These are distinctly different activities, even though the object of their attentions can be common. Journalists and others often use the term "science" as a convenient shorthand to include "engineering," but it verbally subsumes engineering into an activity whose fundamental objectives are of another kind altogether. This use of "science" essentially keeps "engineering" out of the vocabulary of children, who consequently do not learn about all the possible ways there are to have fun with shoelaces, neckties and so much more—including real towers, bridges, automobiles, airplanes, power plants, computers, and everything designed and made.)

An after-school snack provides further opportunities for children to learn that design means that there is not just one way to do something. Consider the problem of designing a method for eating an Oreo cookie with a glass of milk. Different children (and adults) employ different techniques. Those with big enough mouths might just pop the whole thing in. Most eat the cookie in steps, some taking a bite at a time, as if it were a real sandwich. Others proceed by first twisting or prying off one side of the cookie to expose the cream. Some eat the separated top right away; others put it aside and attack the cream first. Even this allows for variations: Some lick the cream off, and others scrape it off with their teeth; some use their top and others their bottom teeth. After finishing the cream, those who put the top aside still have another choice to make: whether to eat the top or bottom next. All along, the glass of milk on the table has allowed for further variations on the process, for the cookie may be dunked or not before each bite. Countless everyday activities, in school or out, provide ample opportunities to introduce young children to design and therefore to engineering.

Invention—within Bounds

Design pervades the lives of children and adults alike; virtually nothing that we do goes untouched by it. We design our own approaches to the everyday things of life, such as lacing our shoes, knotting our ties and eating our cookies. But we also design our own procedures for washing our hands, taking a shower, putting on our clothes. As I recall, in one episode of *All in the Family*, Archie Bunker's son-in-law, Mike, watches Archie put on his shoes and socks. Mike goes into a conniption when Archie puts the sock and shoe completely on one foot first, tying a bow to complete the action, while the other foot remains bare. To Mike, if I remember correctly, the right way to put on shoes and socks is first to put a sock on each foot and only then put the shoes on over them, and only in the same order as the socks. In an ironic development in his character, the politically liberal Mike shows himself to be intolerant of differences in how people do common little things, unaccepting of the fact that there is more than one way to skin a cat or put on one's shoes.

At times we do prescribe how certain everyday things are done, even though there might be countless ways to vary the procedure. This is especially the case in more formal social situations, where doing things too individually might detract from the formality or, in some instances, even prove to be repulsive to polite society. Thus we have manners and social protocols. Arbitrary as they sometimes seem, such things as table manners and restraint in creativity at the table obviate distractions that otherwise might make eating with others, especially strangers, a less than pleasant experience. Imagine a business lunch where the group of people around the table ate with the individuality that children show when eating cream-sandwich cookies. As many ways to eat a sandwich as there might be, there are also practical reasons beyond decorum for following a customary procedure. By keeping the sandwich intact and bringing it to the mouth in the conventional way, we demonstrate one of the sandwich's design features: The fingers are kept free of mustard and mayonnaise, which in turn means that the outside of the drink glass remains relatively tidy throughout the meal and that after lunch the business associates can shake hands without feeling they are washing dishes.

We discover as children, sometimes with the guidance of an adult but often by our own devices, preferred ways to proceed with all sorts of social and recreational activities. There are many ways to design a ball game, and the plethora includes the supplemental use of bats, rackets, bases, baskets, goalposts, nets and more. But when two or more people participate in any game, they must agree on which implements to allow and which rules to follow. Otherwise what transpired would hardly be a game as we know it. Imagine a player on a tennis court serving a football with a baseball bat across a volleyball net to an opponent with a lacrosse stick. Only an agreed-upon set of rules is likely to produce a recreational activity that is not chaotic. If the objective is to have a friendly, or even a fiercely competitive, game, it must proceed according to rules of a rigid design. Even the game of solitaire is only truly played by sticking to the rules. Engineers must certainly stick to the rules of physics, chemistry and the other sciences.

Putting together a jigsaw puzzle is an activity that can be done alone or in a group. Either way, it provides another fine example of how many ways there are to achieve an objective—forming a single picture out of hundreds of pieces of various colors and shapes. Theoretically, it is possible to solve the puzzle by arbitrarily choosing a piece and then trying to fit each of the remaining pieces to it. Systematically trying each piece in each orientation on each side of the starter piece would lead eventually to a match, and the procedure could be followed to completion. I know of no one who works jigsaw puzzles in this tedious and unimaginative way, however, because one of the implicit challenges is to finish the puzzle as efficiently as possible. Most people look for edge and corner pieces first, completing the periphery before tackling the more amorphous middle. If nothing else, this way there are fewer pieces to contend with. As many ways as there might be to complete a puzzle, the preferred way is the most efficient way.

So it is with engineering. There are many ways to build a water crossing, from arranging a set of stepping stones to constructing a majestic bridge or a tunnel. What kind of bridge or tunnel might be best for a given crossing depends on many factors, including river bottom conditions, shipping requirements and traffic capacity. The different ways in which a bridge alone could be designed and constructed are virtually countless, but the added constraint of economy usually makes very few viable. Experienced engineers know which kind of bridge works best for what conditions, just as experienced game players know effective strategies for winning and experienced puzzle solvers know what pieces of the puzzle are best attacked first. Everyone benefits from experience, but we must often rely on the experience of others to get our start in a new endeavor. This is certainly true when students are looking ahead to career choices.

Making Engineering Evident

Children used to see possibilities for their lives in the familiar roles of cowboy, nurse or teacher. Today they may more readily think in terms of astronaut, athlete or rock star. Everyone visits a doctor now and then, which also provides exposure to a common career goal. Many young people learn about other options through family and friends, who often serve as role models. And a good number must rely on what they are exposed to in school, depending on the experience of teachers to set forth the broader possibilities available in school and beyond. No matter the mentor, engineering will not necessarily be seen as an option. It likely depends on how the idea of design is perceived and presented by teachers and parents alike.

Middle or high school children are often introduced to design in the context of "science projects" that are really "engineering projects." Among the most common is the bridge-building contest, in which students are asked to make a model bridge out of balsa wood, Popsicle sticks, spaghetti or some such fragile material. Even though the contest is often associated with a science course, the students are seldom given any substantive guidance about how to visualize, let alone calculate, the structural forces involved. Most necessarily proceed by imitation of bridges they have seen in pictures or across highways. Increasingly, student-friendly computer programs have become available, most notably the West Point Bridge Designer (see http://bridgecontest.usma.edu), in which students can design virtual bridges and test them on the screen. Seldom, however, are such contests presented as exercises in engineering as opposed to, say, applied physics.

Before the collapse of the World Trade Center twin towers, live on television and replayed over and over on videotape, the most widely known failure of a large engineering structure was that of the Tacoma Narrows Bridge. This 1940 disaster was captured on film, but it was subsequently transferred to numerous other media and has been shown to generations of high school students, usually in their physics class as an example of resonant vibration caused by the wind. In fact, the collapse mechanism is much more complicated (described in this column in September–October 1991), and represents an example of wind-structure interaction. Although sophisticated distinctions are understandably absent from most high school physics courses, this case does offer an opportunity to say a bit about the bridge as an artifact of engineering design. Instead of focusing exclusively on the bridge's final dramatic writhing as an illustration of a physical principle, some background on the design of the bridge as an engineering achievement, albeit flawed, would introduce students to a profession that they might find appealing for the opportunities that it presents to change the world for the better.

It is also a familiar middle or high school assignment for students to build small vehicles powered solely by the energy stored in a rubber band or a mousetrap spring. These too are really engineering design problems, but they are seldom presented as such. Rather, at best they are presented as applications of physics, and at worst as mere competitions to devise the machine that travels fastest or farthest. There is certainly nothing wrong with students enjoying the race, but it is unfortunate indeed if the pedagogical opportunity is missed to introduce the joys of design and to inform students that they are engaging in engineering, something that they might spend their lives enjoying—if only they take enough math and science to satisfy admission requirements for engineering school.

Teachers cannot be faulted for failing to promote engineering if they have not been exposed to it themselves. Engineering is not taught in every teacher's college, and it is not a required field of study even in most full-service universities. It is certainly possible to get a bachelor of arts or science—and a teaching certificate—without appreciating that engineering is a profession as noble, rewarding and satisfying as medicine or law. The absence of even the playful rudiments of engineering in the curriculum is unfortunate, as I have learned from doctors and lawyers who have expressed disappointment that they were not exposed more to engineering while in school themselves.

I compare engineering design to making sand castles or lacing up shoes or eating cookies or designing toys not to trivialize it but to humanize it. The conventional wisdom, among the general population as well as among many teachers of children, is that engineering is a cold, dehumanizing and unsatisfying career. Those who hold such a view are not likely to have met or spoken with engineers who enjoy what they do. They are no longer children playing with blocks or building castles on the beach, of course, but many of them retain a certain childlike fascination with the elemental structure of the world and with what can be done with timber and concrete and steel—or with atoms and molecules and microbes. They know that what they have fun designing and building and overseeing is essential to the smooth working of civilization. We should all learn this as children.

_ Henry Petroski

Acknowledgment

This essay was prompted by an invitation to give the keynote luncheon address at the Children's Engineering Workshop cosponsored by the National Aeronautics and Space Administration and Christopher Newport University and held in Williamsburg, Virginia, on January 24.

Computer Science Is an Engineering Discipline

Michael C. Loui
University of Illinois at Urbana-Champaign

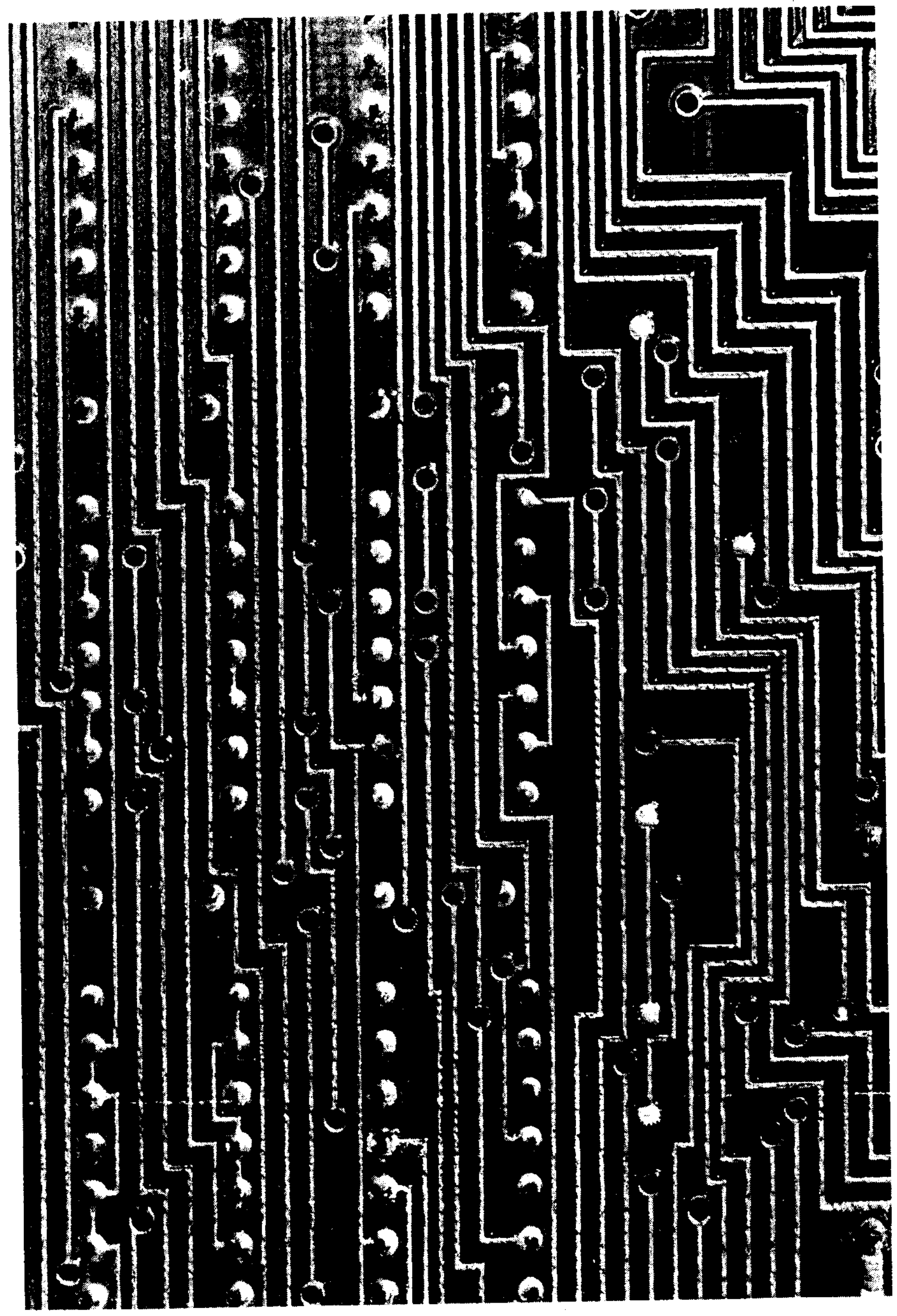
The author asserts that since it is impossible to define a reasonable boundary between computer science and computer engineering, they are the same discipline.

Is computer science an intellectual discipline worthy of representation as an academic department in a university? In an essay widely circulated among mathematicians, Steven Krantz of Penn State wrote, "Computer science is . . . not an end in itself: it is a tool. . . . Last week's news is this week's trivia. The subject is all hardware and few ideas."[1] Perhaps the hostile attitude of many academics arises from misconceptions about computer science and ignorance of its intellectual foundations. I shall briefly review the body of knowledge that makes computer science a genuine intellectual discipline.

What Is Computer Science?

A popular misconception is that computer science studies the use of computers. Those who harbor this misconception argue, rightly, that universities should not grant academic degrees in computer science to people taught only to use computers, any more than they should grant degrees in toaster-oven science to people taught to use toaster-ovens. Further, no intellectual discipline of lasting value can be built on skill in the use of a specific machine that could become obsolete in 20 years.

This misconception arises from identifying computer science with its artifact, the electronic digital computer. The intellectual discipline of computer science has a body of con-



cepts and principles independent of commercially available computers and that have little relevance to the use of computers.

Computer science is the theory, design, and analysis of algorithms for processing information, and the implementations of these algorithms in hardware and in software. The processing of information includes storage, transformation, retrieval, and transmission of information. Within computer science are six interrelated areas:

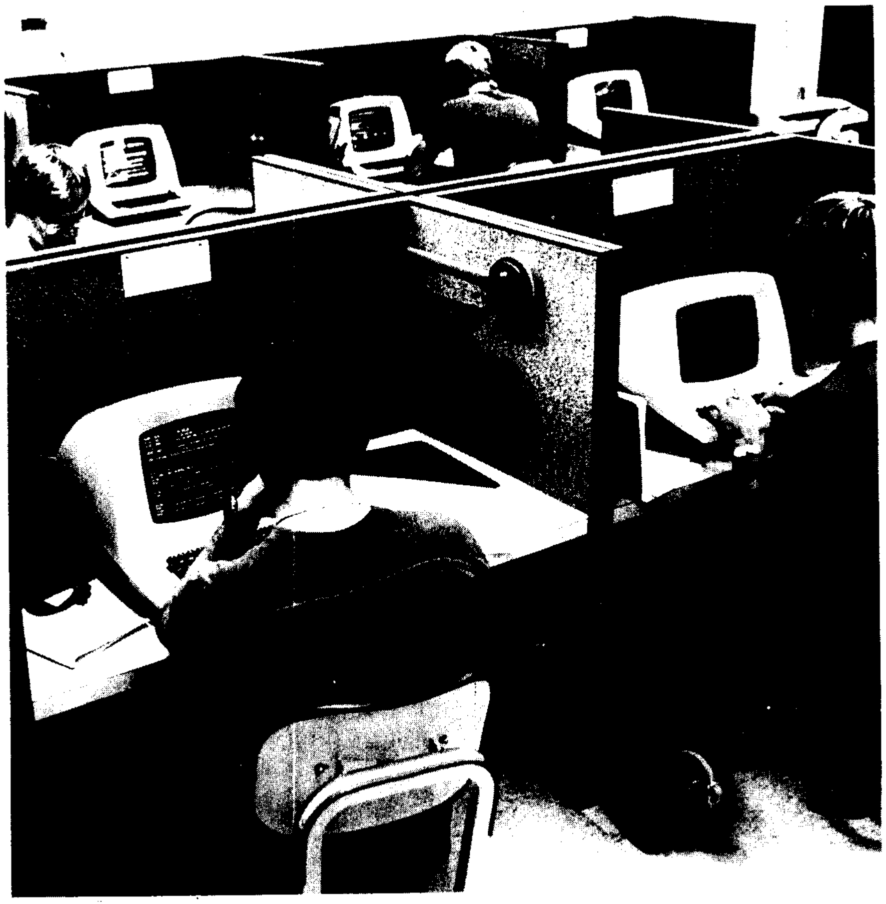
1) Theory of computation (e.g., analysis of algorithms);
2) Software (e.g., programming languages and compilers);
3) Hardware (e.g., logic design);
4) Resource management (e.g., operating systems);
5) Computational mathematics (e.g., numerical analysis);
6) Artificial intelligence (e.g., natural language understanding).

Others have divided the computing field in a somewhat different way.[2]

Like other intellectual disciplines, computer science has a corpus of concepts and principles. Important concepts include the abstract data type, which specifies information mathematically in terms of operations on the information, analogous to defining an algebraic group in terms of its axioms; the abstract data type separates specification from implementation. Important principles include the equivalence of pushdown automata and context-free languages. Several groups have proposed influential model curricula[3-5] that organize the fundamental concepts and principles of computer science into pedagogically appropriate sequences. The emergence of these curricula and of standards for accrediting curricula[6] bespeak the maturity of computer science as an intellectual discipline.

Computer Science As an Engineering Discipline

Is computer science an engineering discipline? One observer who does not seem to think so has written:



The fundamental difference between, say, physics and computer science is that in physics, we study to a very large extent a world that exists.... Computer science, on the other hand, is primarily interested in what can exist.... There is a substantial engineering component in computer science (or its applications), particularly in building computing machines and managing large software projects, but its core activities do not fit the traditional engineering paradigms.[11]

But it is precisely the emphasis on "what can exist" that makes computer science an engineering discipline. Its core activities may not fit traditional engineering paradigms because computer science is an entirely new kind of engineering.

Computer science has all the significant attributes of engineering: a scientific basis, application of principles in design, analysis of trade-offs, and heuristics and techniques. Let us consider these attributes one at a time.

1) *Scientific Basis.* Unlike traditional engineering disciplines such as mechanical engineering and electrical engineering, the fundamental concepts and principles of computer science are not rooted in the physical phenomena of force, heat, and electricity, but in mathematics. Since its scientific foundations are mathematical instead of physical, computer science is a new kind of engineering.

2) *Application of Principles in Design.* The ultimate goal of an engineering project is a product such as a steel bridge or a computer program that benefits society. Like a structural engineer, who applies the principles of mechanics to design a bridge, a computer specialist applies the principles of computation to design a digital system or a program. (The designer may apply these principles either consciously and directly or unconsciously and indirectly.) Most engineers recognize the development of a digital system as an engineering activity. The development of a large program is also an engineering activity, for it involves problem analysis, specification, design, testing, implementation, and maintenance—all classic engineering tasks.

"

Despite its nontraditional mathematical basis, computer science qualifies as an engineering discipline because it has all the characteristics of engineering.

"

"Software engineering" is an accepted term for the development of large software systems.

3) *Analysis of Trade-offs.* The analysis of trade-offs is a salient characteristic of engineering. To implement algorithms efficiently, the designer of a computer system must continually evaluate trade-offs between resources: running time vs. memory space, performance vs. cost, hardware vs. software. For example, hardware implementation of an operation usually runs faster but is harder to change, while software implementation runs more slowly but is easier to change.

Engineering has been defined by its heuristics.[12] Computer science has many heuristics. For example, the Principle of Spatial Locality states that the next memory location accessed is usually near the present location. A popular heuristic of programming methodology is that a program module should be about one page long. Some heuristics have become sufficiently well understood to be called techniques. Important techniques of computer science include pipelining, buffering, recursion, and linked addressing.

Despite its nontraditional mathematical basis, computer science qualifies as an engineering discipline because it has all the characteristics of engineering. Indeed, at several major universities such as Cornell and Stanford, the computer science department is in the engineering college.

Computer Science vs. Computer Engineering

Computer engineers typically claim that computer science concerns only the applications of computers, or only theory, or only software, and computer engineering comprises everything else. Computer scientists frequently claim that computer engineering concerns only hardware, and computer science encompasses everything else.

The applications of computers to a discipline should be considered properly a part of the natural evolution of the discipline. For example, numerical weather forecasting is the province of meteorology, not of computer science. The mass spectrometer has permitted significant advances in chemistry, but there is no "mass spectrometry science" devoted to the study of this instrument.

The theory of computation may seem remote from the engineering of computers. John Doyle has classified the theory of computation as a kind of applied mathematics separate from computer engineering, analogous to the division between rational mechanics and mechanical engineering.* During the twentieth century, however, all engineering disciplines have matured by strengthening their scientific foundations. Excluding the theory of computation from computer engineering would be as inconceivable as excluding circuit theory, linear system theory, communication theory, control theory, electromagnetic theory, and semiconductor theory from modern electrical engineering.

Computer engineering should include software just as computer science should include hardware. Since software development is an engineering activity, software engineering belongs to computer engineering. To enforce a dichotomy between hardware as computer engineering and software as computer science would obscure the fundamental principle

*Personal communication.

What Computer Science Is Not

Computer Science Is Not Mathematics. Although the principles of computer science are mathematical, computer science is not mathematics. Mathematics includes precise statements about objects, but computer science includes precisely specified algorithms for computing objects. As Abelson and Sussman have explained, "Mathematics provides a framework for dealing precisely with notions of "what is." Computation provides a framework for dealing precisely with notions of "how to."[7]

Despite striking analogies between mathematical proofs and computer programs,[8] mathematics and computer science use proofs differently. In mathematics, proving a theorem is a social process[9] in which mathematicians rely on informal, high-level intuitions. In computer science, proving the correctness of a program involves detailed applications of formal inference rules.

Computer Science Is Not Electrical Engineering. Some electrical engineers assert that since a computer is merely a digital system, computer science is properly a part of electrical engineering. I doubt that these engineers would accept a similarly specious reductionist argument that since a transistor is merely impure semiconductor material, electronics is properly a part of chemistry. The emphasis on the computer as a digital system misses the point of computer science: the principles of computation are independent of specific technological realizations in, say, semiconductor microelectronics. These principles would remain the same for computers built out of mechanical components instead of electronic components. "The computing scientist could not care less about the specific technology that might be used to realize machines, be it electronics, optics, pneumatics, or magic."[10]

that hardware and software are functionally equivalent.

This discussion has been summarized:

Computer science includes in one discipline its own theory, experimental method, and engineering. This contrasts with most physical sciences, which are separate from the engineering disciplines that apply their findings—as, for example, in chemistry and chemical engineering. I do not think the science and the engineering can be separated within computer science because of the fundamental emphasis on efficiency.[13]

It is impossible to define a reasonable boundary between the disciplines of computer science and computer engineering. They are the same discipline.

Computer Science and a Liberal Education

Many liberal arts colleges have created computer science curricula, treating computer science as one of the natural sciences. Misunderstandings arise because the engineering orientation of computer science is incompatible with the sciences. For instance, computer science laboratories emphasize engineering design, whereas biology laboratories emphasize measurement and observation. How can colleges reconcile education in computer science with a liberal education, to which professional engineering education seems antithetical?

Undergraduate programs in engineering, including computer science, need not train professional engineers, any more than undergraduate programs in psychology or geology train professional psychologists and geologists. Some liberal arts colleges, including Dartmouth and Princeton, have offered strong undergraduate engineering programs for many years, teaching engineering in a humanistic way similar to the more traditional liberal arts.

Although computer science is an engineering discipline, colleges and universities can design computer science programs to meet the intellectual and pedagogical goals of a broad liberal education.[14]

Summary

Computer science is a true intellectual discipline. Although computer science is not mathematics, its concepts and principles are mathematical, and every day a flourishing research community augments its foundations. Although computer science is not electrical engineering, it is a new kind of engineering, the engineering of algorithms for processing information.

Acknowledgments

This essay has benefited from the constructive criticisms of Edward S. Davidson, Jon Doyle, J. Craig Dutton, Glen G. Langdon Jr., Peter W. Sauer, and Timothy N. Trick.

References

1. Krantz, S.G., "Letter to the Editor," *American Mathematical Monthly*, vol. 91, no. 9, Nov. 1984, pp. 598-600.

2. Sammet, J.E. and A. Ralston (eds.), "The New (1982) *Computing Reviews* Classification System," *Communications of the ACM*, vol. 24, no. 1, Jan. 1982, pp. 13-25.

3. ACM Curriculum Committee on Computer Science, "Curriculum '78: Recommendations for the Undergraduate Program in Computer Science," *Communications of the ACM*, vol. 22, no.3, March 1979, pp. 147-166.

4. *The 1983 IEEE Computer Society Model Program in Computer Science and Engineering*. IEEE Computer Society Press, Silver Spring, Md., 1983.

5. Shaw, M. (ed.), *The Carnegie-Mellon Curriculum for Undergraduate Computer Science*, Springer-Verlag, New York, 1985.

6. Mulder, M.C. and J. Dalphin, "Computer Science Program Requirements and Accreditation," *Computer*, vol. 17, no. 4, April 1984, pp. 30-35. Also *Communications of the ACM*, vol. 27, no. 4, April 1984, pp. 330-335.

7. Abelson, H. and G.J. Sussman, *Structure and Interpretation of Computer Programs*, MIT Press, Cambridge, Mass., 1985.

8. Gries, D., *The Science of Programming*, Springer-Verlag, New York, 1981.

9. De Millo, R.A., R.J. Lipton, and A.J. Perlis, "Social Processes and Proofs of Theorems and Programs," *Communications of the ACM*, vol. 22, no. 5, May 1979, pp. 271-280.

10. Dijkstra, E.W., "Mathematicians and Computing Scientists: The Cultural Gap," *Mathematical Intelligencer*, vol. 8, no. 1, 1986, pp. 48-52. Also *Abacus*, vol. 4, no. 4, 1987, pp. 26-31.

11. Hartmanis, J., "Observations About the Development of Theoretical Computer Science," *Annals of the History of Computing*, vol. 3, no. 1, Jan. 1981, pp. 42-51.

12. Koen, B.V., "Toward a Definition of the Engineering Method," *Engineering Education*, vol. 75, no. 3, Dec. 1984, pp. 150-155.

13. Denning, P.J., "What is Computer Science?" *American Scientist*, vol. 73, no. 1, Jan.-Feb. 1985, pp. 16-19.

14. Gibbs, N.E. and A.B. Tucker, "A Model Curriculum for a Liberal Arts Degree in Computer Science," *Communications of the ACM*, vol. 29, no. 3, March 1986, pp. 202-210.

Michael C. Loui is associate professor of electrical and computer engineering at the University of Illinois at Urbana-Champaign. His research interests include computational complexity theory and the theory of parallel and distributed computation. He received a 1985 Dow Outstanding Young Faculty Award from ASEE.