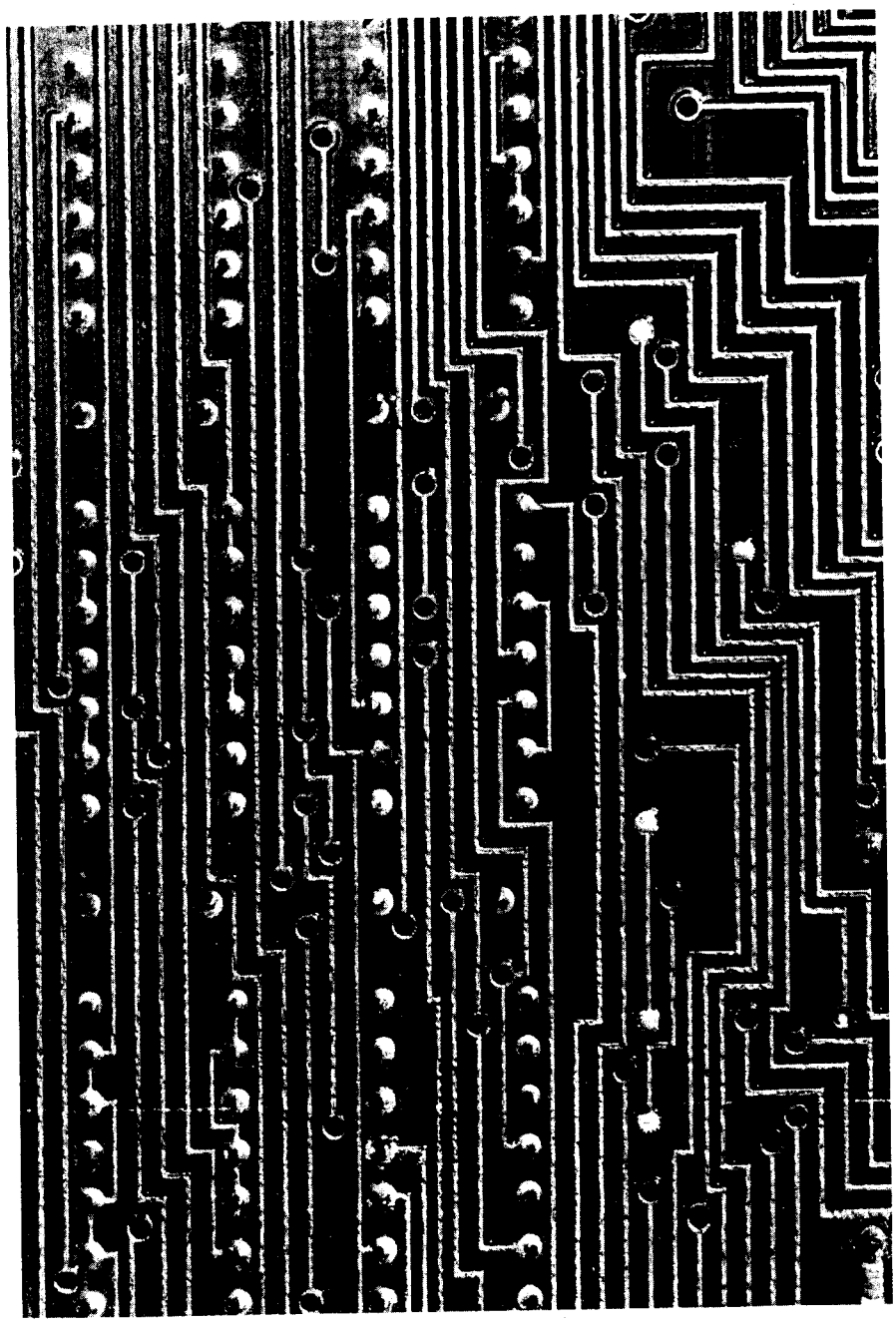
Computer Science Is an Engineering Discipline

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:

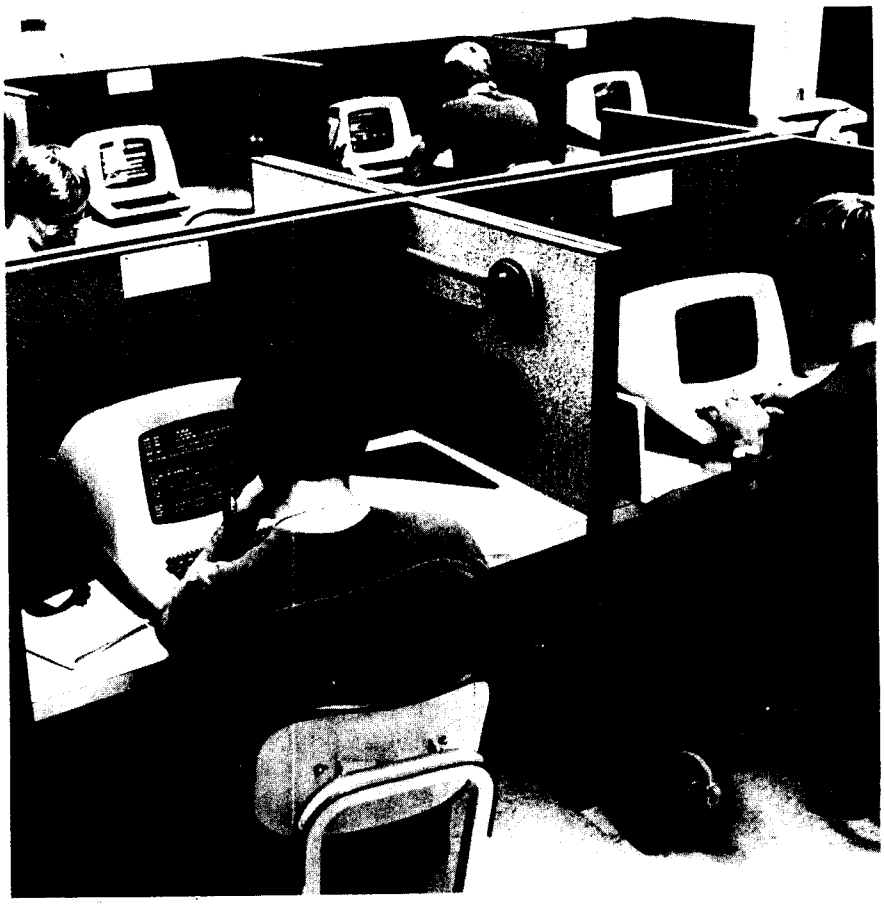
1) Theory of computation (e.g., analysis of algorithms);
2) Software (e.g., programming languages and compilers);
3) Hardware (e.g., logic design);
4) Resource management (e.g., operating systems);
5) Computational mathematics (e.g., numerical analysis);
6) Artificial intelligence (e.g., natural language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