ILL: 39910060	Call Number: sel L 001.609 A613	DateReq: 2/20/2008
	Location:	Date Rec: 2/26/2008
	Maxcost: $30.00IFM	Borrower: BUF
Request Type:	Source: ILLiad	LenderString: IEC,ORU,*INU,JHE,JHE
OCLC Number: 24344740	Affiliation:	

[ ] Yes
[ ] No
[ ] Conditional

Staff Email:

Billing Notes:

Title: IEEE annals of the history of computing.

Uniform Title:

Author:

Edition: Imprint: Los Alamitos, CA : IEEE Computer Society, 1992-

Article: P. Ceruzzi: Electronics technology and computer science 1940-1975: a coevolution

Vol: 10 No.: 4 Pages: 265-270 Date: 1988 or 1989

Dissertation:

Verified: <TN:387758><ODYSSEY:128.205.243.243/LML> OCLC 1058-6180

Borrowing Notes:

ShipTo: ILL/234 LOCKWOOD LIBRARY/SUNY AT BUFFALO/BUFFALO, NY 14260

E-delivery Addr: (716)645-3721 ;ARIEL 128.205.111.1

Ship Via: Library Rate

ShipVia: Library Rate

Return To:

Northwestern University Library /Interlibrary Loan
1970 Campus Drive
Evanston, IL 60208-2300
United States of America

Ship To:

ILL
234 LOCKWOOD LIBRARY
SUNY AT BUFFALO
BUFFALO, NY 14260

NeedBy: 3/21/2008

Borrower: BUF

ILL: 39910060 Lender: INU

Req Date: 2/20/2008 OCLC #: 24344740

Patron: Rapaport, William

Author:

Title: IEEE annals of the history of computing.

Article: P. Ceruzzi: Electronics technology and computer science 1940-1975: a coevolution

Vol.: 10 No.: 4

Date: 1988 or 1989 Pages: 265-270

Verified: <TN:387758><ODYSSEY:128.205.243.243

Maxcost: $30.00IFM Due Date:

Lending Notes:

Bor Notes:

Annals Hist Comput (1989) 10:257–275

Electronics Technology and Computer Science, 1940–1975: A Coevolution

PAUL CERUZZI

This paper explores the relationship between two disciplines: electrical engineering and computer science, over the past 40 years. The author argues that it was the technology of electronics—the exploitation of the properties of free electrons—that finally permitted Babbage's concepts of automatic computing machines to be practically realized. Electrical Engineering (EE) activities thus "took over" and dominated the work of those involved with computing. Once that had been done (around the mid-1950s), the reverse takeover happened: the science of computing then "took over" the discipline of Electrical Engineering, in the sense that its theory of digital switches and separation of hardware and software offered EE a guide to designing and building ever more complex circuits.

*Categories and Subject Descriptors: K. 2. [**Computing Milieux**]: History of Computing—hardware, software, systems, theory. A.1. [**General Literature**]: Introductions and Survey.*
General Terms: Design, Reliability, Theory.
Additional Terms: Computer Science, Electrical Engineering.

Introduction

In 1976, a colorful brochure put out by the IBM Corporation had a startling title: "It Was to Have Been the Nuclear Age. It Became The Computer Age: the Evolution of IBM Computers" (Figure 1).[1] Leaving aside the question whether it is proper to identify any period of time by a piece of technology, the title does call attention to the fact that the computer seems to have sprung up suddenly and unexpectedly, to dominate much of the nation's technology, economy, and culture.

But a sophisticated description of a digital computer appeared in the writings of Charles Babbage in the 1830s. Why, then, the appearance of a "computer age" in the past three decades, and not sooner? A complete answer to this question would include a mix of economic and social as well as technical factors. This essay focuses on an aspect of the internal development of computer technology that was as important as any: namely that after 1940, Babbage's conceptual formulation of an computer was joined to another technology that was well suited to its realization. That technology was electronics.

Electronics emerged as the "technology of choice" for implementing the concept of a computing machine between 1940 and 1955. As it did so, it enabled persons not trained in Electrical Engineering to exploit the power and versatility of computers. This activity led to the study of "computing" independently of the technology out of which "computers" were built. In other words, it led to the creation of a new science: "Computer Science."

The term "coevolution" implies that there was a continuous and reciprocal interaction between electronics and computing. Such interaction did,

[1]An earlier version of the paper was presented to the annual meeting of the Society for History of Technology (SHOT), October 23, 1986, Pittsburgh, Pennsylvania.

Author's Address: Dept. of Space Science and Exploration, National Air and Space Museum, Smithsonian Institution, Washington, D.C. 20560. (202) 357-2828.

in fact occur. As computer science matured, it repaid its debt to electronics by offering that engineering discipline a body of theory which served to unify it above the level of the physics of the devices themselves. In short, computer science provided electrical engineering a paradigm, which I call the "digital approach," which came to define the daily activities of electrical engineers in circuits and systems design.[2]

Though continuous, the interaction between Computer Science and Electrical Engineering was marked by two distinct phases. In the first phase, between 1940–1955, electronics took over the practice of computing. In the second, from 1955 to 1975, computing took over electronics. I shall look at each in turn.

Between 1940 and 1950, a scattered group of persons, without knowledge of one another, put Babbage's ideas into working machinery. These inventors were interested in building machines that could carry out a sequence of elementary arithmetic operations, store intermediate results, and recall those results automatically as needed, and display or print the final results of the calculation. They were not, for the most part, concerned with the engineering details of their implementation, except insofar as they wished to have a machine that worked reliably (Cohen 1985). As things turned out, the first reliable, working computers—in other words, the first machines to implement Babbage's idea of an automatic computing machine—used relays or similar electromechanical elements to carry and manipulate numbers. Using relays (a technology borrowed from the telephone industry), George Stibitz of Bell Laboratories and Konrad Zuse of the Henschel Aircraft Company in Berlin each built calculators that could carry out three to five arithmetic operations a second. And using a combination of relays and toothed wheels borrowed from punched-card accounting machines, Howard Aiken at Harvard built a powerful "Automatic Sequence Controlled Calculator" with a similar operating speed (Ceruzzi 1981).

Relay computers played the vital role of introducing the concept of automatic, sequential calculation to an often skeptical community. It was with electromechanical relay technology that the first automatic calculators, finally, after years of hope and promise, came into existence. But almost from the start they were eclipsed by machines using the much faster vacuum tube as its computing element. The story of the invention of the electronic digital computer has been told elsewhere, and in that story the issue of the vacuum tube's perceived unreliability, as well as its heavy power demands, are among the difficulties cited for the initial skepticism as to its practicality. These were indeed serious issues, but they were addressed. Once they were, vacuum tube technology, with its higher operating speeds, was perceived as an alternative to relays.

One reason for the rapid ascendancy of electronic devices for computing elements was that events during the war, mainly unrelated to building computers, had transformed electronics itself, raising it above the level considered (and rejected) by the computer pioneers like Aiken or Stibitz. One development—radar—was critical, and became the bridge across which electronics entered the realm of computing.

The role of radar is not usually considered as

Paul Ceruzzi was born in Bridgeport, Connecticut, and attended Yale University, where he received a B.A. in 1970. He attended graduate school at the University of Kansas, from which he received his Ph.D. in American Studies in 1981. His graduate studies included a year as a Fulbright Scholar at the Institute for the History of Science in Hamburg, West Germany, and he received a Charles Babbage Institute Research Fellowship in 1979. Before joining the staff of the National Air and Space Museum, he taught History of Technology at Clemson University in Clemson, South Carolina.

*Dr. Ceruzzi's main scholarly work has been in the history of computing since 1935. His has written a book on this subject (**Reckoners, the Prehistory of The Digital Computer, 1935–1945**, Greenwood Press, 1983), and he is presently working on a major new gallery at the National Air and Space Museum about the computer's impact on air and space flight.*

[2]Throughout this paper I will be concentrating on that branch of Electrical Engineering that is more accurately described as "electronic" engineering. This term will be defined later in the text, but essentially I will not address that branch of EE that deals with Power Engineering or the so-called "Heavy Currents."

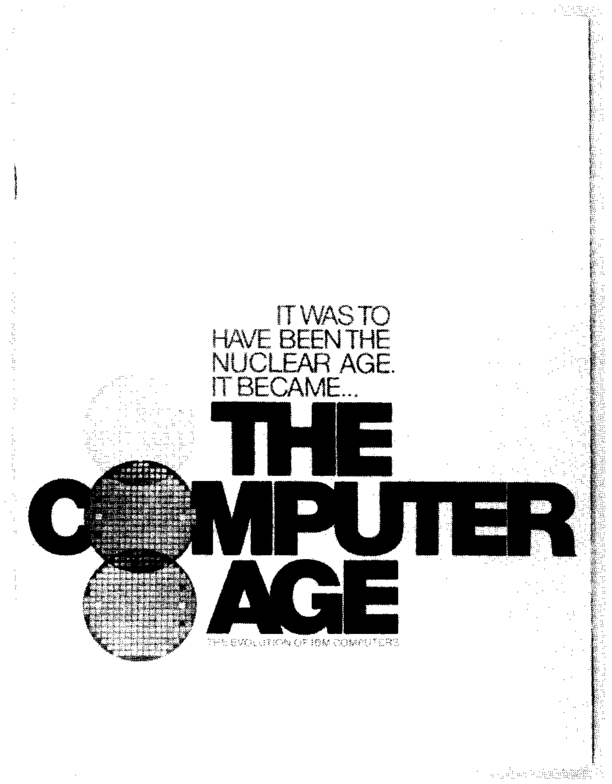

Figure 1. Brochure from IBM, undated, about 1976 (IBM Corporation).

part of the generational lineage of computer history, in contrast to, say, the invention and marketing of mechanical adding machines. Yet among those involved in modern computing's first decade, there was no question as to its influence. Radar required vacuum tube circuits that handled discrete pulses of current at high frequencies, in contrast to radio transmitters and receivers which had been the mainstay of prewar electronics engineering. Both these requirements matched the needs of computer engineering. Radar sets typically contained over a hundred tubes, again in contrast to the more modest four- or five-tube radio sets of the day.

One link from radar to the computer was the mercury delay line: an ingenious and tricky device developed with some difficulty for storing radar pulses. After the war those who had experience with it (e.g. Maurice Wilkes at Cambridge University in England and Presper Eckert in Philadelphia) could adapt it for use as a computer memory device. Those who were less familiar with it had less success, many (e.g. Aiken) believing that such a device was so fragile that it would never work in a computer (Wilkes 1985, p. 128). Mercury delay lines were indeed difficult to build and operate; nevertheless they played the role of being the memory device for four of the first five stored program computers to be built in the United States and England: EDSAC, BINAC, SEAC, and Pilot ACE (the exception was the Manchester computer, later called "Mark I," which used a Williams-tube memory.[3]

In 1953, when the *IRE Proceedings* issued a special "Computer Issue," Werner Buchholz, the guest editor, stated that although many computer projects were started during WW II,

> . . . Still, the present growth of the computer industry did not start until the results of the enormous development of electronic technology during World War II were brought into the field. It is interesting to note that many computer projects started around a nucleus of wartime radar experts. Electronics not only provided the technological means for greatly increased speed and capacity, and thereby enhanced the usefulness of computers many times, but the availability of cheap, mass-produced components and of engineers trained to use them made it possible to experiment on a greater scale and at a lower capital investment than before (IRE 1953, p. 1220).

As a result of that development of electronics technology between 1939 and 1945, employment in the American electronics industries had risen from 110,000 to 560,000 (*Electronics* 1980, pp. 150–210).

The vacuum tube ascendancy was not immediate, however. Experience with radar had attacked many of the problems of reliability, but these problems still remained. Just as serious was the fact that the much faster operating speeds of tubes required a rethinking of the overall structure of a computing machine, especially the way it received its instructions. High electronic speeds meant nothing if the computing circuits received their orders by mechanical devices such as paper tape or punched card readers. Likewise the high arithmetic speeds had to be carefully matched to equally high speeds for storing and retrieving intermediate results from memory devices. It was also recognized that higher arithmetic speeds required greater memory capacities. Each of the first electronic calculators (i.e., machines whose computing program was not directed by a stored pro-

[3]See Table 1 for a full listing of these and other early computers and their memory devices.

gram), namely the Atanasoff computer, the ENIAC, the British Colossus, and the IBM SSEC, addressed these problems in different, and in hindsight inelegant, ways. Electronic computing was held up by the need for a consensus on what a digital computer ought to look like.[4]

This last bottleneck was broken in 1945, with the emergence of the concept of the stored program principle as the way to organize the various units of a computer. The origins of this concept are a matter of controversy, but the informal distribution, in 1945 and 1946, of a "First Draft of a Report on the EDVAC" by John von Neumann was what brought the computing community a general awareness of the concept (von Neumann 1945).

Von Neumann's report described the EDVAC in terms of its logical structure, using a notation borrowed from neurophysiology. The EDVAC's implementation in vacuum tube circuits, though mentioned, is not the focus of von Neumann's energies. Instead he focuses on the main functional units of the computer—its arithmetic unit, its memory, input and output, and so on. The report also described the idea, and the advantages, of storing both instructions and data in one, high speed internal memory.

The "First Draft" had an effect on every aspect of computing. One effect was to hasten the demise of the relay and assure the place of electronic circuits as the technology of choice for building a computer. Once the logical design of a digital computer was laid out in a way not tied to a specific technical implementation (as von Neumann's was), then it became no more difficult to construct a computer according to that design using vacuum tubes than it was to construct it out of relays or anything else. There were problems of reliability with tubes, but these were not overwhelming, nor were they that much greater than similar problems (e.g. transient faults) with relays (Stibitz 1945). What that meant was that for an incremental investment of time and money to utilize vacuum tube technology, one got a thousandfold increase in speed. That advantage was overwhelming, and it meant that the argument of tubes vs. relays was over before it had a real chance to begin.

Maurice Wilkes, whose EDSAC was among the first stored program computers to be completed, in 1949, was among those who saw the Report as the answer to many of the organizational problems associated with building computers:

> . . . In [the EDVAC Report], clearly laid out, were the principles on which the development of the modern digital computer was to be based: the stored program with the same store for numbers and instructions, the serial execution of instructions, and the use of binary switching circuits for computation and control. I recognized this at once as the real thing, and from that time on never had any doubt as to the way computer development would go (Wilkes 1985, p. 109).

As the first stored program electronic computers finally began operating in the early 1950s, their superiority was quickly recognized. The Aberdeen Proving Ground provided a good test environment—at that facility a variety of mechanical and electronic calculators, relay sequence calculators, and stored program computers were installed by 1953. Franz Alt, who was at Aberdeen at that time, later remarked:

> . . . relay computers were in competition with them [electronic computers], and they didn't hold their own. They were much too slow by comparison . . . After a few years people lost interest in them. They had been built as an insurance against the possibility that electronic computing might not work (Alt 1969).

The range and evolution of machine types, and the emergence of the stored program approach, is revealed by the listing in Table 1 of digital computer installations, broken down by their type of design. (Analog computers and devices are examined in a separate section). Table 1 summarizes the various types of automatic computing machines built and installed between 1940 and 1955. For each machine, a date is given for its completion or first installation, followed by an

[4]It is sometimes argued that those who were skeptical of vacuum tube technology because of its alleged unreliability were correct, as evidenced by the fact that vacuum tubes were eventually themselves replaced by the presumably more reliable "solid-state" devices such as diodes, transistors, and later on integrated circuits. Transistors did offer far greater reliability and lower power consumption than tubes, but it was a full decade after the transistor's invention that it became practical to produce transistors in quantity with uniform characteristics such that they could be used in computers. In the invervening decade (1948–1958), it was questions first of logical structure, and then of memory technology, that dominated debates over computer design.

Table 1. Computer Installations, 1940–1955.

Name	Year	No. Installed
I. Electromechanical and Electronic Calculators		
A. Relay or Mechanical Calculators		
Bell Labs Model 1	1940	1
Zuse Z-3	1941	1
Bell Labs Model 2	1942	1
Harvard Mark I	1944	1
Bell Labs Model 3	1944	1
Bell Labs Model 4	1945	1
Zuse Z-4	1945	1
Bell Labs Model 5	1946	2
Harvard Mark II	1947	1
ARC (see text)	1948/52	1
Bell Labs Model 6	1949	1
ONR/ERA relay computer "Abel"	1950	1
BARK	1950	1
ARRA	1952	1
NEC Mark I (Tokyo)	1952	1
B. Electronic Calculators, General Purpose but Externally Programmed		
IBM PSRC	1944	5
ENIAC	1945	1
IBM 603 Multiplier	1946	100
IBM 604	1948	5600 eventually
IBM SSEC	1948	1
Harvard Mark III	1949	1
Northrop Aircraft/IBM CPC	1949	700
Harvard Mark IV	1952	1
ERA Logistics Computer	1953	1
Burroughs E-101	1955	100
Monrobot	1955	5 approx.
Elecom 50	1955	2
C. Special Purpose Electronic Calculators		
Jaincomp	1950	4
USAF Fairchild	1950	1
OMIBAC	1950	1
Teleregister SPEEDH	1952	>4
Reservisor	1952	1
BAEQS	1953	1
Magnefile	1954	>3
TRADIC (Bell Labs)	1954	1
MDP-MSI	1954	>1
Haller Ray & Brown	1955	1
MIDSAC		1
D. Digital Differential Analyzers		
Northrop MADDIDA	1949	15
CRC 101 & 105	1951	>5
QUAC	1952	1
Bendix D-12	1954	2
Wedilog		1

Table 1. Computer Installations, 1940–1955. (*continued*)

Name	Year	No. Installed
II. Stored Program Electronic Computers		
A. Serial, "EDVAC type"		
BINAC	1949	1
EDSAC	1949	1
SEAC	1950	1
Pilot ACE	1951	1
RAYDAC	1951	1
UNIVAC	1951	46 eventually
C.S.I.R.O. Mark I (Australia)	1952	1
CUBA (France)	1952	1
EDVAC	1952	1
LEO	1952	1
DYSEAC	1953	1
FLAC	1953	1
MIDAC	1953	1
DEUCE	1954	3 by 1955, 32 eventually
B. Drum Memory, "ERA 1101 type"		
ERA 1101	1950	1
OARAC	1952	1
Burroughs Laboratory Comp.	1952	5 approx.
CADAC 102	1952	14
Elecom 100	1952	5 approx.
PTERA	1952	1
Bendix G-15	1953	>400 eventually
CALDIC	1953	1
CIRCLE	1953	5
Hughes Airborne	1953	5 approx.
IBM 650	1953	>2000 eventually
MINIAC	1953	3
ALWAC	1954	5
ORDFIAC	1954	1
WISC	1954	1
PENNSTAC	1955	1
LGP 30	1955	>100 after 1955
READIX	1955	1
C. Parallel Memory, "von Neumann type"		
Whirlwind	1950	1
SWAC	1950	1
Manchester (Ferranti) Mark I	1951	9
AVIDAC	1951	1
IAS (von Neumann)	1951	1
ILLIAC	1952	1
IBM 701	1952	19 by 1955
MANIAC	1952	1
NAREC	1952	1
ORDVAC	1952	1
ARC (see text)	1948/52	1
ERA 1103	1953	10
JOHNNIAC	1953	1
IBM 702	1954	14
Ferranti Mark II	1954	19
NORC	1954	1
ORACLE	1954	1
IBM 704	1955	1 in 1955, many later

estimated number of installations (in many cases this information is approximate).[5]

The figures verify Alt's impressions of this era: relay calculators and computers initiated the digital era, but they were quickly eclipsed by electronic devices. Of the electronic machines, general purpose calculators having a limited degree of programmability (e.g. Northrop/IBM CPC, IBM 603/604) were installed in large numbers in the early part of this period, and served as the workhorse of digital computation until inexpensive stored program computers became available, beginning around 1953. Special purpose electronic machines, including the Digital Differential Analyzer, likewise fall into this category.

Stored program electronic computers did not begin to appear in large numbers until around 1953, especially with the introduction of the IBM 650. Those that were installed in large numbers tended to be the slower, but less expensive, drum types (such as the 650). These numbers should be considered in the context of the greater speed, memory capacity, and overall computing power of the large scale machines such as the UNIVAC and IBM 701, of which only a few were installed in the early years.

The "Von-Neumann type" design (stored program, parallel memory access), which provided the fastest performance, and which became the standard architecture down to the present day, was slow in being established in the form of installed machines, even though its advantages were widely known through a series of reports by von Neumann and others at the Institute for Advanced Study after 1946. This was mainly due to the fact that the viability of the parallel architecture depended on a reliable, fast, and relatively cheap memory device; something which did not really become commercially available (in the form of magnetic cores) until the end of this era.

Perhaps symbolic of this phase of the history of the computer was the ARC computer, built by A. D. Booth of London's Birkbeck College, and first operational in 1948. Booth had followed the American developments closely, and was convinced early on of the advantages of the von Neumann, stored program approach to computer design. He proceeded to design and build a computer along these lines, however using relays instead of vacuum tubes in the interests of saving money and time. But almost as soon as the ARC was completed, he set out replacing the relay circuits with their equivalent vacuum tube circuits to implement the same logical functions (Booth 1949).

Beginnings of Computer Science, 1955–1975

Between 1955 and 1975, a science of computing, in North America adopting the name "Computer Science," emerged. Its focus was on the electronic, stored program digital computer (with its magnetic core memory) invented in the previous decade. As this new science emerged, the electronics technology from which it sprang changed in response. The first change was a steady progression of computer related papers in the Electronics journals.[6] Eventually it affected the very definition of "electronics" itself. Before examining the emergence of a science of computing after 1955, consider the accepted definition of "electronics" at that time.

The many changes in the practice of electronics during the Second World War, of which computing was but one, led electrical engineers to reexamine the nature of the discipline. The original definition of electronics was that of the movement of electrons in a gas or vacuum, and was intended to distinguish radio and communications work from the power engineering out of which Electrical Engineering first emerged (McMahon 1984). This definition stemmed from Edison's observation in 1883 that an evacuated tube could carry a current, and from the inven-

[5]This table is a summary of a report that has been compiled from various sources, primarily the ONR Digital Computer Newsletter, and the Ballistic Research Laboratory's *Survey of Automatic Computers,* three volumes of which were also published during this interval (Weik 1955). The complete listing, with references for each machine, is on file at the Computer Museum, Boston, and the Charles Babbage Institute, Minneapolis. In looking at this table it is important to recognize that the definition of computing machine was not constant during that period, to the extent that a machine that might appear on the table as a "computer" in 1945 would not qualify as a "computer" by 1955. However in all cases I have sought to include machines that went at least a step beyond performing simple arithmetic on a pairs of numbers, but which could with some degree of automatic control evaluate mathematical expressions of at least a modest length.

[6]There is a steady increase of computer related papers, as indexed in *Science Abstracts, B* (Electrical Engineering), from zero in 1945 to 10% in 1965. In 1960, computer papers were divided into analog and digital, with analog papers dropping off to about 2% in 1965. After 1965, as argued later in this paper, digital computing began to dominate *all* electronics papers, so that the frequency of computer papers indexed is no longer a measure of its dominance in the field.

tion of the diode and triode vacuum tubes in the first decade of the 20th century.[7]

With the advent of servomechanisms, radar, computers, and the transistor (which did not involve movement of electronics though a vacuum), the definition had to change. In a guest editorial in a 1952 issue of the *IRE Proceedings*, William Everitt proposed a new definition:

> Electronics is the Science and Technology which deals primarily with the supplementing of man's senses and his brain power by devices which collect and process information, transmit it to the point needed, and there either control machinery or present the processed information to human beings for their direct use (Everitt 1962, p. 899).

In subsequent issues, several readers objected, saying that the notion of "information" was too vague. Many felt that Everitt was correct in broadening it beyond the movement of electrons in a vacuum, but they suggested that a better, yet still precise, definition might be something along the lines of "the movement of electrons, in solid, gas, or vacuum (McMahon 1984, pp. 231–232)."

The increasing awareness of the computer as a machine that integrates all aspects of information handling, including communication, was implied in a 1959 address by Simon Ramo, of Hughes Aircraft, to the Fifth National Communications Symposium, where he proposed a new term, "Intellectronics," defined as "the science of extending man's intellect by electronics (Ramo 1959)."[8] And Zbigniew Brzezinski coined the term "Technetronics" to describe essentially the same transformation of society (Brzezinski 1970).

By 1977 the computer-and-information definition had become accepted, at least as a general overall definition of electronics, as indicated by the lead article by John Pierce for a special issue of *Science* on "the Electronics Revolution":

> What is electronics? Once we associated electronics with vacuum tubes, but vacuum tubes are almost obsolete. Perhaps electronics is semiconductor devices. But then, what of magnetic cores and bubbles and liquid crystals? I think that electronics has really come to mean all electrical devices for communication, information processing, and control . . . (Pierce 1977).[9]

Recent publications hint at a new definition that is in the same spirit as Everitt's 1952 definition of electronics as a matter of communication and control. Mainly as a result of the development of so-called very large scale integration (VLSI)—integrated circuits with hundreds of thousands of elementary devices on one chip—there is a perception that it is the job of electrical engineers to "manage complexity." Although it is still of concern to design the elementary transistors, resistors, and so on, what is now the critical issue is how to interconnect thousands, even millions of similar and fairly simple devices to one another.

In the final chapter of Karl Wilde's and Nilo Lindgren's book on the history of Electrical Engineering and MIT, for example, several current faculty and administrators are asked to define what they see as the essence of their department. Most agree that the computer, and especially its implementation in VLSI circuits, had come to dominate the practice of the electrical engineer.[10] For one of the administrators interviewed (Fernando Corbato), "if there is a single theme . . . it is the problem of complexity (Wildes and Lofgren 1985)."[11] This last definition, if it becomes generally accepted, reveals a strong in-

[7]Shortly after Edison's observation, J. J. Thompson explained the effect by hypothesizing that a stream of negatively charged particles carried the current through the vacuum. In 1894 these particles were given the name "electrons"—hence "electronics"—by the Irish physicist George Stoney. Alan Turing, whom I shall describe later in this paper as one of the founders of Computer Science, was a distant relative of Stoney. (Hodges 1983).

[8]Ramo's term did not catch on, although a decade later a group of engineers working in what has since become known as "Silicon Valley" founded a company called "Intel," a contraction of either Ramo's term or of the words "Integrated Electronics;" (Hanson 1982, Chapter 5). Today, what in America is known as "Computer Science" is called "Informatics" in Continental Europe.

[9]Notice that although the definition of electronics-as-information resolved the question of whether solid-state devices like the transistor and integrated circuit properly belonged to electronics, it introduced the confusion of allowing electromagnetic relay devices to be included. While in the general view there is nothing wrong with this inclusion, when applied to the context of the early digital computer era, it is inappropriate.

[10]They further agreed that the decision made in the late 1970s to keep Computer Science as a part of EE and not let it break off as did many other universities was a wise one; the name of the department is now Electrical Engineering and Computer Science. I shall have more to say on this later.

[11]One example might serve to illustrate the increase in complexity of electronic circuits that has taken place over the past few decades: When the World Trade Center was built in Lower Manhattan beginning in the late 1960s, it displaced a block known as "Radio Row": a group of shops selling surplus radio parts, mainly of World War II vintage. Though many lamented the passing of this area, it should be noted that in terms of active circuits, the entire contents of *all* the shops on Radio Row are today contained on one or two VLSI chips.

fluence from the theory and practice of computing.

Definition of "Computer Science"

While the definition of electronics had been changing, a new term appeared, a term whose definition would also evolve: "Computer Science." Though it is now one of the most popular subjects taught in universities, "Computer Science" has had a variety of definitions.[12] Its history is brief, and any statements as to what it is become dated quickly. Nevertheless there are general areas of agreement as to its nature.

One definition, stated in its extreme, is that it is not a science at all. (In debates found in the letters column of computer trade and professional journals, one sometimes reads the aphorism, "Any science that needs the word 'science' in its name is by definition not a science.") A less extreme form of this statement is that computer science is driven by electronics technology, and that computer scientists do little more than observe and collate into general rules the behavior of the machines the engineers supply them. Though useful, such rules are not scientific principles such as those on which Electrical Engineering itself was based (e.g. Maxwell's theory of electromagnetism, semiconductor physics). As such, computer science is not a true science but one of the "engineering sciences," in Edwin Layton's terms, which "took on the qualities of a science in their systematic organization, their reliance on experiment, and in the development of mathematical theory" (Layton 1971). Computer science's rules and laws tend to be about observable and practical things, such as the time or the memory requirements for a certain program that sorts a file, and not about the fundamental properties of computing, whatever they may be.

This perception has its historic roots in the 1945–1955 period, when it took heroic engineering efforts to get a computer to work at all. The feeling was that any talk about a theory of computers was premature when it is questionable whether one could get a stored program computer to run without failure for more than a few minutes at a time. (The one exception to this was of course the "theory" of the stored program as stated in von Neumann's EDVAC Report.) This attitude survives among contemporary computer engineers, who although often too young to know of the difficulties of electronic computing's first decade, are nonetheless paid only to get a machine "out the door" of the factory. Other than having the machine pass certain benchmarks to validate its performance, they are not concerned with what the customer (including the computer scientists) do with it (Kidder 1981).

One of the first general textbooks on electronic computers stated flatly that "The outstanding problems involved in making machines are almost all technological rather than mathematical (Bowden 1953). As computer science matured, others noted that for at least a century there had been a tradition in a branch of mathematics of studying the notion of a mechanistic process, a tradition that began with George Boole's *Investigation into the Laws of Thought* . . . , published in 1854, and which included the work of Frege, Hilbert Gödel and many others on the formalization of mathematical expressions (van Heijenoort 1967; Aspray 1980). But this tradition" . . . was smothered after 1940 by a great technological explosion" (Wegener 1970). In the famous Moore School lectures held in 1947, shortly after the public unveiling of the ENIAC, there was almost no mention of any of these men and women—despite the fact that the title of the lectures was "Theory and Techniques for Design of Electronic Digital Computers" (Moore School 1947). In particular, the tone of the Moore School sessions was that any discussion of the theory of computer design had to take a back seat to the pressing technical problem of designing a fast and reliable memory system.[13] (The stored program computers then being built used for their memory either mercury delay lines or specially built television tubes. Both methods had extreme limitations, especially regarding reliability, cost per

[12]A recent survey of colleges and universities shows that in 1983, 25,000 bachelor's degrees were awarded in Computer Science, compared with 18,000 in Electrical Engineering, 11,000 in Chemistry, 12,600 in Mathematics and Statistics, and 3800 in Physics (Gries et al. 1986).

[13]For example, Claude Shannon's work, which showed that the rules of symbolic logic were well suited as a design tool in the construction of relay circuits that performed arithmetic, was hardly mentioned at all. For many of the participants at these lectures the important thing was to find out what kind of hardware was inside the many "block diagrams" the lecturers kept putting on the blackboards. As noted above, the one exception was the theory of computer design as described by von Neumann, but even in this instance there was a feeling at the sessions that theoretical design for the EDVAC was being emphasized too much, in the place of a more narrative description of the ENIAC, a computer based on an ad hoc theory but one that was at least functioning in 1947.

bit of storage, and capacity (Redmond and Smith 1980).

A few years later, a few companies and universities had succeeded in building working computers. Among them there was a modest debate over the value of logical state diagrams as a guide to computer design, as opposed to straight engineering borrowed from radar circuits that handled electrical pulses. For a while the former was known as the "West Coast" approach to computer design, but by the mid 1950s the debate fell by the wayside as computer design finally established itself on a firmer foundation of symbolic logic (Sprague 1972).

Another indication of how technology drove perceptions of computing comes from the way the history of computers was marketed. When IBM introduced the System/360 series of computers in 1964, they helped promote the notion of computers belonging to three "generations," defined according to the technology by which they were implemented: vacuum tubes, transistors, and integrated circuits. This classification, which has since become accepted and even expanded on (viz. the Japanese "Fifth Generation" project), had at least two effects on the perception of the history of computing: first, it relegated all computer activities before the ENIAC into a limbo of either "prehistory" or irrelevant prologue; second, it defined progress in computing strictly in terms of the technology of its hardware circuits.

Recently this view of computer science being technology driven has been repeated by C. Gordon Bell, for many years chief of engineering for the Digital Equipment Corporation and one of the inventors of the minicomputer. Speaking of the "invention" of the personal computer in the late 1970s, he said:

> A lot of things are called inventions when, actually, they were inevitable. I believe technology is the driving devil. It conspires, and if there's a concept half-there or a computer half-designed, technology will complete it (Bell 1985).

Elsewhere, he has said of the computer industry:

> It is customary when reviewing the history of an industry to ascribe events to either market pull or technology push. . . . The history of the computer industry . . . is almost solely one of technology push (Bell et al. 1978).

And in Tracy Kidder's chronicle of the team of young engineers in 1978 who were bidding a new computer for Data General:

> Some engineers likened the chips to an unassembled collection of children's building blocks. Some referred to the entire realm of chip design and manufacture as 'technology,' as if to say that putting the chips together was something else. A farmer might feel this way: 'technology' is the new hybrid seeds that come to the farm on the railroad, but growing those seeds is a different activity—it's just raising food (Kidder 1978, p. 122).

After 1955 there arose compelling arguments that computer science was a genuine science, albeit one that differed in many ways from the classical sciences. The first argument to appear was one that emerged in response to the pressure that computing activities were putting on traditional disciplinary boundaries, especially in the universities. With one exception (Wiesner, noted below), the first published statements about "computer science" revealed a perception that a science was being born, and it needed to be established at least on organizational and administrative grounds; the question of just what it "was" could be answered later. By the late 1950s it was recognized that many topics that had much in common with each other (and all in common with the computer) were being taught in various departments around most universities. The feeling was that those who were concerned with the computer aspects of their work in these other departments would perhaps not be recognized and adequately rewarded by their peers for doing good work. Establishing a separate department of "computer science" would address that concern.

By the second volume (1959) of the *Communications of the ACM* (the flagship journal for the Association for Computing Machinery), the term "computer science and engineering" had begun to appear. That September, an article entitled "The Role of the University in Computers, Data Processing, and Related Fields," by Louis Fein, discussed the need to consolidate, under a single organizational entity, the various studies orbiting around the computer in various academic departments such as business and economics, mathematics, linguistics, library science, physics, and electrical engineering. After mentioning a few names for this entity, including "information sciences" (which he attributed to Jerome Wiesner), "intellectronics" (which he says was suggested by Simon Ramo), and "synnoetics" (which Fein himself had suggested elsewhere) he suggested "the 'computer sciences'"; later in the article shortened to "Computer Science," (singular, and in quotations). This I believe is the origin of the term (Fein 1959; Fein 1961; Leech 1986).

In describing what this new discipline was, Fein made the further point about what it was NOT:

> "Too much emphasis has been placed on the computer equipment in university programs that include fields in the 'computer sciences' . . . Indeed an excellent integrated program in some selected fields of the computer sciences should be possible *without* any computing equipment at all, just as a first-rate program in certain areas of physics can exist without a cyclotron" (Fein 1959, p. 11).

The establishment of Computer Science's administrative and organizational boundaries was followed five years later by the first attempts to establish it as a true science based on its internal nature. These attempts centered on the concept that, like any other, Computer Science is the systematic study of certain phenomena, only in this case the object of study is an artificial, not a natural one. In other words, Computer Science is simply the study of computers. It is not to worry that computers are artificial and not natural phenomena. This was the argument of Herbert Simon, author of *The Sciences of the Artificial,* and of his colleagues Allen Newell and Alan Perlis, then at Carnegie–Mellon University, whose letter to the editor of *Science* in 1967 was the first explicit statement to this effect (Newell et al. 1967).

For Newell, Perlis, and Simon, Computer Science is the study of computers, just as Astronomy is the study of stars. But in Astronomy, stars will be stars, no matter what the astronomer says about them. This is not so with computers. If what the computer scientist says about computers in theory does not agree with observed behavior, he or she can always change the computer (more correctly, get an electrical engineer to change the computer). For most of the period 1950–1980, there was sufficient continuity of the von Neumann, stored program model of computer architecture that this did not present a problem.[14] As long as the basic architecture remains constant, so too will the definition emphasize the non-hardware aspects of computing, such as in the 1976 *Encyclopedia of Computer Science*:

> Computer Science is concerned with information processes, with the information structures and procedures that enter into representations of such processes and their implementation and information processing systems. . . . The central role of the digital computer in the discipline is due to its near universality as an information processing machine. With enough memory capacity, a digital computer provides the basis for modeling any information processing system, provided the task to be performed by the system can be specified in some rigorous manner. . . . the stored program digital computer . . . provides a methodologically adequate, as well as a realistic, basis for the exploration and study of a great variety of concepts, schemes, and techniques of information processing (Amarel 1976).

Since the publication of the Newell et al. letter to *Science*, a different internal definition has emerged, and it is this one which dominates the day-to-day practice of computer science today, especially in the universities. That is the definition of computer science as the study of algorithms—effective procedures—and their implementation by programming languages on digital computer hardware. Implied in this definition is the notion that the algorithm is as fundamental to computing as Newton's Laws of Motion are to physics; thus Computer Science is a true science because it is concerned with discovering natural laws about algorithms which are not engineering rules of thumb or "maxims," in Layton's terms (Layton 1971, p. 566; Vincenti 1979). Computer Science thus becomes a science because it has a theoretical foundation on which to build, such as: "The notion of a mechanical process and of an algorithm (a mechanical process that is guaranteed to terminate) are as fundamental and general as the concepts that underlie the empirical and mathematical sciences (Wegener 1970, pp. 70–78)." It is no coincidence that this theoretical foundation is essentially based on the work of Hilbert, Turing, Church, and others whose pre-1940 work in mathematics was neglected by those building the first electronic computers of the 1940s.[15]

The event that, more than any other, gave the algorithmic basis currency was the publication, in 1968, of a book entitled *Fundamental Algo-*

[14]Lately, with the development of so-called parallel multiprocessors, things have changed, and there is a corresponding change in the attitude of computer science that increasingly sees hardware issues as relevant. We may now see a return to the pre-1950 era when hardware issues dominated discussions of the theory of computing. I discuss this issue further in a later section of this paper.

[15]A few texts in Computer Science assert that the fundamental principle of Computer Science is the so-called *Turing–Church Hypothesis*, which, informally stated, says that all algorithmic procedures are equivalent to a class of mathematical functions known as general recursive functions (Aspray 1980, Ch. 2). At the same time it should be noted that few if any textbooks in Computer Science devote much space to an elaboration of this hypothesis.

rithms by Donald Knuth (Knuth 1968).[16] Intended as the first of a planned seven-volume series on "The Art of Computer Programming," Knuth's book was a conscious effort to place computer programming on a foundation of mathematical principles and theorems. The book sparkled with wit and erudition. It was a specialized text that many nonspecialists could read and enjoy. (One of its most memorable passages was a 13-page, detailed analysis of the algorithms implied in getting an elevator in the Mathematics Building at the California Institute of Technology to work correctly. Although the elevator itself worked fine, Knuth later recognized with a shrug that his algorithmic analysis contained several fatal "bugs.") Throughout the book, Knuth codified and formalized a wealth of computing techniques and tricks that had been informally known for years among computer programmers.[17]

Another of Knuth's strengths was his command of history. A large portion of each volume discussed the historical context of computing; indeed, Knuth was one of the first to recognize and appreciate the advances in mathematics (*computational* mathematics) made by the ancient Babylonians and by Europeans during the Middle Ages. These historical passages, far from being diversions from the main thrust of the text, gave strong support to the notion that "computer programming," if defined properly, was part of a long scientific tradition.

Fundamental Algorithms consciously defined the study of algorithms as a subject that was independent of any machine that might implement them. Recognizing the need for a programming language to describe algorithms of interest throughout the book, Knuth deliberately introduced "MIXAL," a language developed especially for the book and one not implemented on any machine.[18]

The notions of a mechanical process and an algorithm, whether Computer Science's first principles or not, did form an informal basis for the way the subject has been taught in universities since the mid-1960s. But a rigorous analysis of the Turing–Church Hypothesis or of Turing's 1936 paper "On Computable Numbers . . . " was rarely found in introductory Computer Science Courses. What was more likely to be found was a smattering of history that included a brief mention of formal computability, quickly followed by the meat of the course: programming in one or more high-level computer languages. In short, Computer Science textbooks adopted Knuth's strategem of using a programming language as a medium for describing algorithms; unlike Knuth, most university and college texts introduced actual programming languages (e.g. FORTRAN, PASCAL), whose programs could and were executed by the students on their university computers.

The present curriculum of the majority of Computer Science programs stems from a series of reports published in the journals of the Association for Computing Machinery, and the curriculum that evolved reveals a trend away from hardware concerns toware more and more of a mathematical basis (as taught, however, the mathematical level of many computer science courses is not high). The version published in 1968 (the so-called "Curriculum '68)" was especially influential; one person calls it comparable to the EDVAC report as a founding document of academic computer science (Pollack 1977). In its first, preliminary version, "electronics" appears as an optional elective under "supporting" courses (Figure 2) (ACM 1965). In "Curriculum '68" hardware courses are gone completely, to be replaced by an algorithmic approach and an emphasis on languages and data structures (ACM 1968). In the latest version the algorithmic focus remains, with more mathematics introduced in early stages (ACM 1977).

A more recent definition has appeared, one that echoes the one thing is emerging in electrical engineering. That is the perception of computer science as the study of complexity, at all its levels. The notion that there is a new science of the management of complexity appeared in one of the

[16]Volumes 2 and 3 of the series (*Seminumerical Algorithms* and *Sorting and Searching*) appeared in 1969 and 1973, respectively. Volumes 4 through 7 have yet to appear. In 1976, Knuth hinted that more volumes were forthcoming, but as of this writing (1988), they have not (Knuth 1976; 1982).

[17]One example from Knuth's third volume in the series may serve to illustrate: "By 1952, many approaches to internal sorting were known in the programming folklore, but comparatively little theory had been developed . . . None of the computer-related documents mentioned so far actually appeared in the 'open literature'; in fact, most of the early history of computing appears in comparatively inaccessible reports" (Knuth 1973, pp. 386–387)."

[18]Shortly after the appearance of the book, however, computer programmers wrote programs, called "cross compilers," that allowed a range of then-existing computers to execute MIXAL statements. But no "MIX" computer, which Knuth called "the world's first polyunsaturated computer . . . very much like nearly every computer now in existence, except that it is, perhaps, nicer," was ever built (Knuth 1968, p. 120).

COMPUTER SCIENCE COURSES

Table of Courses for Computer Science Majors

TABLE 1. Preliminary Recommendations of the Curriculum Committee of ACM for Majors in Computer Science

Courses / Recommendations	Computer Science: *Basic Courses*	Computer Science: *Theory Courses*	Computer Science: *Numerical Algorithms*	Computer Science: *Computer Models and Applications*	*Supporting*
Required	1. Introduction to Algorithmic Processes* 2. Computer Organization and Programming 4. Information Structures	5. Algorithmic Languages and Compilers	3. Numerical Calculus (or Course 7)		Beginning Analysis (12 cr.) Linear Algebra (3)
Highly Recommended Electives	6. Logic Design and Switching Theory 9. Computer and Programming Systems		7. Numerical Analysis I 8. Numerical Analysis II		Algebraic Structures Statistical Methods Differential Equations Advanced Calculus Physics (6 cr.)
Other Electives	10. Combinatorics and Graph Theory	13. Constructive Logic 14. Introduction to Automata Theory 15. Formal Languages		11. Systems Simulations 12. Mathematical Optimization Techniques 16. Heuristic Programming	Analog Computers Electronics Probability and Statistics Theory Linguistics Logic Philosophy and Philosophy of Science

* The specifications for the 16 Courses, indicated by boldface numerals, are given in this report.

Figure 2. Preliminary curriculum from the ACM's Committee on Computer Science, 1965. [Note the single course on "electronics" under the heading of "supporting" and "other electives." (from ACM Curriculum Committee on Computer Science, "An Undergraduate Program in Computer Science: Some Recommendations, *Communications ACM* 1965, p. 546).]

first published statements about what we now call "computer science." In an address at a ceremony opening the IBM San Jose Laboratory in 1958, Jerome Wiesner made the following comments:

> Information processing systems are but one facet of an evolving field of intellectual activity called communication sciences. This is a general term which is applied to those areas of study in which interest centers on the properties of a system or the properties of arrays of symbols which come from their organization or structure rather than from their physical properties; that is, the study of what one MIT colleague calls 'the problems of organized complexity' (Wiesner 1958).[19]

Currently this definition's most vocal proponent has been Edsger W. Dijkstra. Beginning in the late 1960s, Dijkstra consistently argued that despite an *apparent* basis on algorithms, Computer Science departments were teaching only engineering rules of thumb about programming languages. For Dijkstra it was (and is) imperative that Computer Science distance itself from not only hardware issues but also from the mastery of programming languages as its principal activity in the schools. His writings, which often take the form of brief notes, serially numbered and privately circulated to his colleagues, echo Wiesner's statement as well as those of MIT's Electrical Engineering Department, for example:

> . . . [N]ow the teaching of programming comprises the teaching of facts—facts about systems, machines, programming languages etc.—and it is very easy to be explicit about them, but the trouble is that these facts represent about 10

[19]The "MIT colleague" is not identified.

percent of what has to be taught; the remaining 90 percent is problem solving and how to avoid unmastered complexity . . . (Dijkstra 1982, p. 107) . . . But programming, when stripped of all its circumstantial irrelevancies, boils down to no more and no less than the very effective thinking so as to avoid unmastered complexity (Dijkstra 1982, p. 163).

For him, the goal of computer science was to concern itself with the attempt "to define programming semantics independently of any underlying computational model . . . " or, in other words, to "forget that program texts can also be interpreted as executable code" (Dijkstra 1982, p. 275). But although Dijkstra is held in high esteem by academic computer scientists, Computer Science as it is taught (especially in the United States) emphasizes the study and mastery of existing programming languages (and operating systems) that can be executed on existing digital computers.

Based on these observations, Computer Science, as it was formally recognized and taught between 1955 and 1975, was an engineering science, according to Layton's term. It also fits Walter Vincenti's criteria for engineering science in that in its first two decades, progress in Computer Science occurred in the absence of any formal or useful theory (Vincenti 1979, pp. 742–746.[20] But in the context of its roots in formal mathematics (and in its ever increasing levels of abstraction and formality since 1975) it is now a pure science, albeit one that is still groping for an agreed upon set of fundamental principles and one that has a different character from classical physics or chemistry.

The issue of what is Computer Science ironically has little to do with its having been shaped by administrative, government, military, and university policies—indeed, the same sort of policy factors are characteristic of nearly all post-World War II science, including (even especially) physics. Computer Science concerns the systematic study of algorithms, especially in the expression of those algorithms in the form of computer programs that can be executed on commercially sold digital computers. Computer hardware, and hence electrical engineering, are part of computer science, but at present the university structure of computer science departments treats computing hardware as a given, and the more one can ignore purely hardware issues the more progress can be made on the study of algorithms. To a lesser extent there is a trend to look at existing programming languages in the same way.

Analog vs. Digital

Despite the many pieces of common ground between computing and electronics, the two activities remained distinct. One reason was due to a fundamental difference in the ways each discipline approached the handling of signals of electron currents, a difference that had characterized the evolution of each discipline from its earliest days. Electrical Engineering evolved as a study of using devices (like the vacuum tube) to amplify continuous signals (McMahon 1984). Computing Engineering, later on Computer Science, was concerned with using electrons to count and switch. The one was analog, the other digital. Analog computing devices, electronic or otherwise, belong to the history of computers (the ENIAC owed as much a debt to wartime analog computing projects as it did to radar or to digital mechanical computing). But analog computers do not belong to Computer *Science*, as the discipline established itself in the late 1950s. The reason is simple: Computer Science centers on the programs that execute algorithmic procedures; but whatever advantages analog have over digital machines, their inability to be programmed easily put them forever at a disadvantage, and preclude their being part of the discipline. Several examples illustrate this difference.

The first concerns the fate of the various Differential Analyzer projects, centered at MIT under the leadership of Vannevar Bush. These machines were certainly among the world's first "computers," in the sense that they were the first machines capable of automatically evaluating fairly complicated mathematical expressions. Yet they never really fulfilled their promise, and of all the reasons this was so, it was the difficulties involved in reprogramming them for different tasks that was decisive (Owens 1986).[21]

[20]In one aspect Computer Science represents a departure from Vincenti's thesis. That is his assertion (Vincenti 1979, p. 746), that ". . . the use of working scale models . . . *is* peculiar to technology. Scientists rarely, if ever, have the possibility of building a working model of their object of concern." As noted by Newell et al. above, the object of study for Computer Science is precisely such a model—a universal model at that.

[21]Owens argues, for example, that it was the Rockefeller Analyzer's inability of to be reprogrammed easily that was

Another example concerns the introduction of digital computing to the aviation industry. This community, consistently one of the largest customers for computer equipment, had a long tradition of using analog computing devices for aircraft stability and control, as well as for ground based applications such as missile tracking and guidance. But at the same time this industry was among the first to adopt the digital approach, as soon as it felt that the problems of reliability and size could be managed. They did so, despite numerous engineering difficulties, again because of the digital computer's greater flexibility (Ceruzzi, 1989).

And at MIT in the early 1950s, a strong research program had developed on automatic control of machinery, especially for automating factory processes and production. This work had its roots in the (essentially analog) engineering of MIT's Servomechanisms Laboratory. By 1952, when a special issue of Scientific American on "Automatic Control" appeared, it was recognized that digital techniques were preferable in all but one aspect. That was the ability of analog devices to operate in "real time." But it was also noted that the steady progress of digital computing indicated that these machines would soon have the speeds necessary for such operations. And when they did, automatic control would be done by digital computers (Ridenour 1952).

Throughout the 1950s and 1960s, advances in digital computer programming permitted its incursion into areas where analog devices had formerly held sway: consider the replacement of the slide rule by the electronic calculator, or the replacement of the engineer's drafting table and machinist's jigs by CAD/CAM (Noble 1984). The final blow, and the event which completed Computer Science's break with Electrical Engineering, was the discovery of the Fast-Fourier Transform, an algorithm which permitted digital computers to tackle signal processing and analysis, a discovery which "thus penetrated the major bastion of analog computation" (Newell 1982).

In short, analog computing faded because no one was able to build an analog computer that had the property of being universally programmable in the sense of a Turing Machine. And in triumphing over analog, digital computing established the notion that the study of the software side of computing was a valid activity, and it validated the departments of Computer Science (not Electrical Engineering) as the places where this study would take place. Computer Science thus emerged and was split from Electrical Engineering between 1940 and 1970 as a result of the resolution of tha analog/digital split in favor of digital.[22]

Computer Science Takes over Electronics

Electronics technology took over computing; after Computer Science established itself, it repaid the debt by taking over electronics. That is, the notion of using electronic components as digital switches, to perform functions that are specified not by their circuit wiring but by "software," came to dominate the activities of the electronics engineer. This notion had its origins in von Neumann's EDVAC Report, and by the end of this period has been formalized by computer scientists. They in turn furnished a paradigm that became an organizing force for the practice of electronics engineering above the physics and engineering of the device level. In other words, in the early decades of computing (1940–1960), the theory of computing drew its conceptual framework from electronics. In the next two decades (1960–1980) computing supplied to electronics a paradigm, namely the digital approach, that has reshaped that discipline.

Consider the following example: the 25 October, 1973 issue of *Electronics* was devoted to "The Great Takeover." The magazine's publisher introduced the issue by saying:

> The proliferation of electronics' multifarious technologies into new products, new applications, and new markets—indeed, into new services never before considered possible—is the theme of this special issue of *Electronics*. On the cover, we have called the pervasive movement of electronics into just about every area of human endeavor "The Great Takeover." And, in many ways it was inevitable, as the cost advantages of

the main cause for its quick demise. Ironically, in grappling with this problem for the Rockefeller Analyzer, Perry Crawford and others at MIT developed their plugboard system of programming for it, a technique which was then applied to the programming of the ENIAC.

[22]The combining of the two in MIT's administrative structure is an isolated case. To the extent it points to a future trend, it is because digital computing has run up against some fundamental limits, including the basic quantum granularity of materials. Therefore progress in computing may require a turning away from the basic principles on which Computer Science has been founded—including, among others, the superiority of digital over analog.

electronic technologies and cost-effectiveness of electronic solutions took over more and more jobs from the venerable mechanical and electromechanical technologies. (*Electronics* 1973, p. 4)

Inside, a 100-page section described in detail how electronic devices were rapidly sending older technologies to the scrap-heap. Examples included retail sales, (where point-of-sale terminals were replacing cash registers), pocket calculators replacing slide rules, electronic circuits replacing mechanical clockwork in watches, computers in banking replacing older posting and accounting techniques, and many others.

Although nowhere was it explicitly stated in this issue, the reason electronics was taking over the world was that digital circuits were taking over electronics. Every example given described a digital circuit. Buried at the end of the section on "technology" was a half-page piece entitled "Don't Forget Linear" (p. 84). The implication was that "linear"—that is, analog—circuits were indeed all but forgotten. And of the linear circuits described, a large percentage were those that performed a conversion between analog and digital.

After about 1973, Electronics Engineering became digital computer engineering. Radio, and communications applications, from which electronics sprang and which dominated it in its earlier period, were still there, but insofar as they were, they were treated as a subset of digital techniques. Even the humble radio, from which modern electronics engineering grew, has now lost its tuning dial to a calculator style digital keypad. Thanks to the mass produced microprocessor, it has become easier to take a computer and program it to "act like a radio," than it is to design and build a radio from scratch.

Analog circuits, now called "linear applications," are still found of course, but they occupy an inferior position. Many do believe however that progress in computer engineering will hinge on a redefinition and breakthrough in analog circuits, as digital circuits approach the physical limits of the ultimate granularity of matter (Sutherland and Mead 1977).[23] Digital computing techniques, and their expression in the microprocessor, offer overwhelming advantages over any other approaches to, say, building a radio or an automatic control system or whatever. Thus it has become not only possible but compelling to recast as much of electronics practice into a digital computing mold. Increasingly, digital computing appears as a natural extension of the very properties of electronics that have always been part of its appealing characteristics.

Conclusion

Electronics took over computing in the late 1940s because of its inherent advantages over other techniques. Digital computing, the theory for which grew out of Computer Science, took over electronics because it provided a path for those inherent advantages to progress. In sum, those advantages are as follows:

Like electronics in general, digital computing offered speed. The "instant" communications offered by the Morse telegraph was matched by the relentless drive by the computer engineer for faster switching circuits, and by the computer scientist's development of algorithms that do not "blow up"—take up exponentially greater numbers of cycles as the complexity of the problem increases by a small increment.

Like electronics in general, digital computing offered leverage. One early notion of electronics (still used in Europe) was that it concerned the applications of "weak currents," in contrast to the "strong currents" of traditional electrical engineering. Weak currents of electricity, carried on thin and light wires or as weightless signals through the ether, do the heavy work of carrying messages, motion pictures, and signals that control heavy machinery. With computers is it the same: a tiny chip and its accompanying ethereal software do the heavy work of "crunching numbers" and moving and processing huge quantities of data.

[23]In recent years computer science has had to return to a closer look at technological questions. This phenomenon is outside the scope of this paper, but briefly it can be summarized as a reaction to the introduction of the microprocessor in 1974, which has driven the cost of computing to near zero. Ivan Sutherland, a founder of computer graphics, and Carver Mead, one of the founders of a theory of VLSI, summed up this phenomenon as follows: "Computer science has grown up in an era of computer technologies in which wires were cheap and switches were expensive. Integrated circuit technology reverses the cost situation, making switching elements essentially free and leaving wires as the only expensive component." (Sutherland and Mead 1977, pp. 210–228). As a result, the theory of computing has to be revised, but as of their writing (1977) this revision had only begun. With the advent of the microprocessor it is now more practical to stamp out very complex computers on a chip, and then deliberately hobble them to do a more mundane task, than it is to design and build from scratch the simpler circuit to do that mundane task.

Finally the digital approach extended the concept in electrical engineering of the separation of a machine's structural design from its implementation. When electric power was introduced into the factory, one of its major advantages was that it allowed the machinery of production to be arranged on the factory floor according to the logic of production, without regard to the logic of the distribution of power as had been the case with mechanical transmission. The design tool of the electronics engineer has been the schematic diagram—unlike, say, the architectural drawing—a diagram that enjoys the luxury of postponing the physical and spatial details of implementation until later. During the Second World War, aviation electronics gave us the "black box"—radio, radar, and control systems whose internal organization were invisible, and irrelevant, to those using it.

From Computer Science came an *explicit* division of a technical process into "hardware" and "software." Progress in the latter depends on the fact that programmers need not be concerned with the details of the former. It is this separation that electrical engineering has siezed upon. By deferring the thorny issues of applications to a later phase of software development, the electrical engineer is better able to cope with the physical problems of constructing circuits having hundreds of thousands of individual active devices, each of which has to mesh with all the others. It is in this sense that the computer scientist has repaid the debt they owed to electrical engineering, by providing a way for the engineer to construct circuits several orders of magnitude more complex than were possible otherwise. Digital engineering has become a paradigm for electrical engineering, in Kuhn's sense of general organization of a body of theory, whose acceptance by a community facilitates their day-to-day work (Kuhn 1970).[24]

But as the paradigm of the stored program digital computer came to pervade the nature of electronics, at the same time important distinctions remained between computer science and electronics technology. The two did not become synonymous, and with the exception of a few institutional settings, they are likely to remain separate academic disciplines (Figure 3). Those who called themselves computer scientists, though their livelihood depended on the existence of computer technology, distanced themselves from hardware issues insofar as possible. They did so in part for institutional reasons, but also because they felt a need to focus their energies on establishing a foundation, based on scientific principles, for the construction and analysis of algorithms that a computer implements. This was an activity they felt would never be considered a proper branch of electronics engineering.

By the end of the 1950s, electronics engineers agreed with computer scientists that their work concerned the processing, storage, and transmission of information, although the definition of "information" was neither precise nor the same for each group. Twenty years later, both groups have begun to see their work as the "manage-

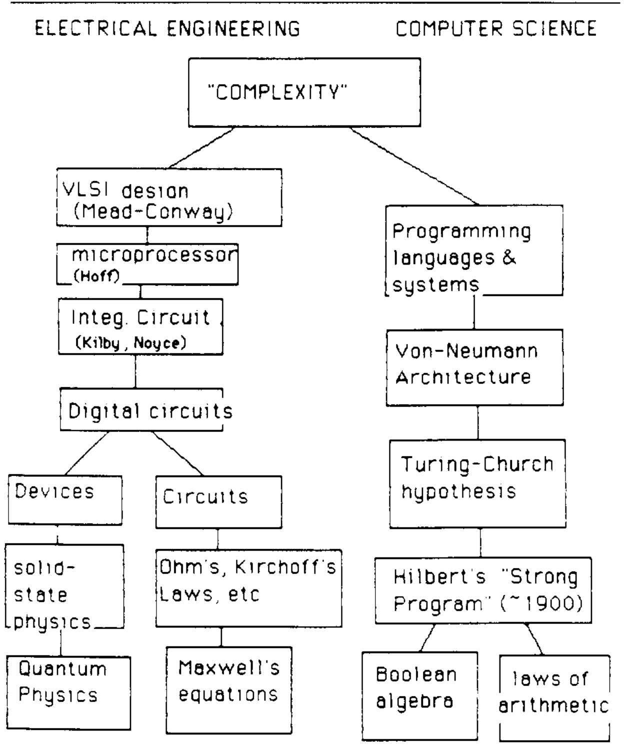

Figure 3. The Merging of Computer Science and Electrical Engineering?

[24]My discussion of the "digital paradigm" for modern Electrical Engineering is based on Kuhn's earliest writings on the subject. The subsequent elaboration and refutation of Kuhn's thesis, by numerous writers including Kuhn himself, does not in my view affect the validity of his original insight as a way of understanding the practice of what he calls "normal science," in this case normal *engineering* science. The distinction between hardware and software is present in all technology, though usually never so easy to discern as in a computer. It is furthermore a distinction which exists in a hierarchy of levels—the microprocessor, for example, is such a versatile circuit because it does not require that its functions be fixed as it leaves the factory. It leaves the electronics engineer free to concentrate on the job of fabricating the chip itself, without having to worry at that time about precisely what it will be used for.

ment of complexity," a term once again not precisely defined. Both disciplines continue to evolve and change rapidly. For both Electronics Engineering and Computer Science, the present definitions, institutional settings, and research programs will evolve, and coevolve, in the future.

Acknowledgments

An earlier draft of this paper was presented at the Annual Meeting of the Society for History of Technology (SHOT), October 1986. The author wishes to thank W. David Lewis, Edwin T. Layton Jr., William Aspray, Michael Williams, Michael Dennis, and Paul Forman for their critical reading and comments on that draft. The Smithsonian Institution's Research Opportunities Fund contributed financial support toward the preparation of this paper.

References

ACM Curriculum Committee on Computer Science. 1965. "An Undergraduate Program in Computer Science: Some Recommendations," *Communications ACM* 8 (September), pp. 543–552.

ACM Curriculum Committee on Computer Science. 1968. "Curriculum 68: Recommendations for Academic Programs in Computer Science." *Communications ACM* 11 (March), pp. 151–197.

ACM Curriculum Committee on Computer Science. 1977. *ACM SIGCSE Bulletin, 9,* pp. 1–16.

Alt, F. 1969. Interview, Smithsonian Computer History Project.

Amarel, S. 1976. "Computer Science," in Anthony Ralston (ed.), *Encyclopedia of Computer Science,* New York: Van Nostrand, pp. 314–318.

Aspray, W. 1980. "From Mathematical Constructivity to Computer Science: Alan Turing, John von Neumann, and the Origins of Computer Science in Mathematical Logic," Dissertation, University of Wisconsin, 1980.

Bell, C. G. 1985, *Computerworld,* Oct. 14, 1985, p. 19.

Bell, C. G., J. C. Mudge, and J. McNamara. 1978. *Computer Engineering: A DEC View of Hardware Systems Design,* Bedford, Massachusetts: Digital Press, p. 27.

Booth, A.D. 1949. "The Physical Realization of an Electronic Digital Computer," *Electronic Engineering 21* (1949), p. 234; *22* (1950), pp. 492–498.

Bowden, B.V. 1953. *Faster Than Thought,* London, Pitman, p. 30.

Brzezinski, Z. 1970. *Between Two Ages: America's Role in the Technetronic Era,* New York: Viking.

Ceruzzi, P. E. 1981. *Reckoners: The Prehistory of the Digital Computer, 1935–1945,* Westport, Connecticut: Greenwood Press.

Ceruzzi, P. E. 1989. *Beyond the Limits: Flight Enters the Computer Age,* Cambridge, Massachusetts: MIT Press.

Cohen, I. B. 1985. Foreword to the 1985 reprint of *A Manual of Operation for the Automatic Sequence Controlled Calculator,* Cambridge, Massachusetts: MIT Press 1985, p. x.

Dijkstra, E. 1982. *Selected Writings on Computing: A Perspective,* New York: Springer.

Electronics, 1980. "Fifty Years of Achievement: A History," *Electronics,* Special Commemorative Issue, 17 April, 1980.

Electronics, 1973. "The Great Takeover," Special Issue of *Electronics,* 25 October.

Everitt, W. 1952. "Guest Editorial," *IRE Proceedings* 40, p. 899.

Fein, L. 1959. "The Role of the University in Computers, Data Processing, and Related Fields," *CACM 2* (Sept.), pp. 7–14;

Fein, L. 1961. "The Computer-related Sciences (Synnoetics) at a University in the Year 1975." *American Scientist, 49*/2 (June), pp. 149–168.

Gries, D., R. Miller, R. Richie, and P. Young 1986. "Imbalance between Growth and Funding in Academic Computing Science; Two Trends Colliding," *Comm. ACM, 29* (September), pp. 870–876.

Hanson, D. 1982. *The New Alchemists: Silicon Valley and the Microelectronics Revolution,* Boston: Little, Brown.

Hodges, A. 1983., *Alan Turing, the Enigma,* New York, Simon & Schuster.

IRE 1953. "Computer Issue." *IRE Proceedings* 41 (October).

Kidder, T. 1981. *The Soul of a New Machine,* Boston: Little, Brown, 1981, pp. 245 ff.

Knuth, D. E. 1968. *The Art of Computer Programming; Vol. 1: Fundamental Algorithms,* Reading, Massachusetts: Addison-Wesley.

Knuth, D. E. 1973. *The Art of Computer Programming; Volume Three: Sorting and Searching,* Reading, Massachusetts: Addison Wesley.

Knuth, D. E. 1976. "The State of the 'Art of Computer Programming,'" Stanford University, Computer Science Dept., Report STAN-CS-76-551, May.

Knuth, D. E. 1982. "A Conversation with Don Knuth," *Annals of the History of Computing, 4*/3 (July), pp. 257–274.

Kuhn, T. 1970. *The Structure of Scientific Revolutions,* 2nd Ed., University of Chicago Press.

Layton, E. T. 1971. "Mirror-Image Twins: The Communities of Science and Technology in 19th Century America," *Technology and Culture,* 12 (October), pp. 562–580.

Leech, J. 1986. Letter to the author.

McMahon, A. Michal 1984. *The Making of a Profession: A Century of Electrical Engineering in America,* New York: IEEE Press.

Moore School of Electrical Engineering, 1947. "Theory and Techniques for Design of Electronic Digital Computers: Lectures Given at the Moore School, 8 July 1946–31 August, 1946." Philadelphia: Moore School of Electrical Engineering, Univ. of Penn., 1947; reprinted 1985, MIT Press, Cambridge, Massachusetts.

Newell, A., A. Perlis, and H. Simon. 1967. Letter to the Editor of *Science* 157 (22 Sept.), pp. 1373–1374.

Newell, A. 1982. "Intellectual Issues in the History of Artificial Intelligence," Carnegie-Mellon University, Department of Computer Science, Report CMU-CS-82-142.

Noble, D. F. 1984. *Forces of Production: A Social History of Industrial Automation* (New York: Knopf).

Owens, L. 1986. "Vannevar Bush and the Differential Analyzer: The Text and Context of an Early Computer," *Technology and Culture, 27* (January), pp. 63–95.

Pierce, J. R. 1977. "Electronics: Past, Present, and Future." *Science, 195* (18 March), pp. 1092–1095.

Pollack, S. 1977. "The Development of Computer Science," in S. Pollack (ed.), *Studies in Computer Science,* Washington, D.C. 1982, pp. 1–51.

Ramo, S. 1959. Address to 5th National Communications Symposium, Utica, NY, 7 Oct. 1959. Quoted in *Computers and Automation,* 9 (Jan. 1960), p. 6.

Redmond, K. C. and T. M. Smith 1980. *Project Whirlwind: The History of a Pioneering Computer,* Bedford, Massachusetts: Digital Press.

Ridenour, L. 1952. "The Role of the Computer," *Scientific American,* Special Issue on "Automatic Control," *187,* (September), pp. 116–130.

Sprague, R. E. 1972. "A Western View of Computer History," *Comm. ACM 15* (July), pp. 686–692.

Sutherland, I. and C. Mead. 1977. "Microelectronics and Computer Science," *Scientific American,* 237 (September), pp. 210–228.

Stibitz, G. 1945. "Relay Computers," National Defense Research Committee, Applied Mathematics Panel, AMP Report 171.1R, February 1945, pp. 21–22.

van Heijenoort, J. 1967. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931,* Cambridge, Massachusetts: Harvard Univ. Press.

Vincenti, W. 1979. "The Air-Propeller Tests of W. F. Durand and E. P. Lesley: A Case Study in Technological Methodology," *Technology and Culture, 20* (October), pp. 712–751.

von Neumann, J. 1945. "First Draft of a Report on the EDVAC," Philadelphia: Moore School of Electrical Engineering, 30 June.

Wegener, P. 1970. "Three Computer Cultures: Computer Technology, Computer Mathematics, and Computer Science," *Advances in Computing,* 10, p. 9.

Weik, M. 1955. *A Survey of Domestic Electronic Digital Computing Systems,* Aberdeen, Maryland: Ballistic Research Laboratory, Report No. 971.

Wiesner, J. 1958. Communication Sciences in a University Environment. *IBM J. Res. Dev. 2*/4 (October), pp. 268–275.

Wildes, K. L. and N. A. Lofgren 1985. *A Century of Electrical Engineering at MIT, 1882–1982.* Cambridge, Massachusetts: MIT Press, 1985.

Wilkes, M. V. 1985. *Memoirs of a Computer Pioneer,* Cambridge, Massachusetts: MIT Press.