The Practice of Informatics

Review ■

The Origins of Informatics

MORRIS F. COLLEN, MD

Abstract This article summarizes the origins of informatics, which is based on the science, engineering, and technology of computer hardware, software, and communications. In just four decades, from the 1950s to the 1990s, computer technology has progressed from slow, first-generation vacuum tubes, through the invention of the transistor and its incorporation into microprocessor chips, and ultimately, to fast, fourth-generation very-large-scale-integrated silicon chips. Programming has undergone a parallel transformation, from cumbersome, first-generation, machine languages to efficient, fourth-generation application-oriented languages. Communication has evolved from simple copper wires to complex fiberoptic cables in computer-linked networks. The digital computer has profound implications for the development and practice of clinical medicine.

■ J Am Med Informatics Assoc. 1994;1:91-107.

The increasing importance of computer applications to medicine should generate some curiosity as to the origins and the evolution of informatics—the science, engineering, and technology of computer hardware, software, and communications. Understanding the historical development of informatics technology can help to elucidate the relevance of informatics to medicine. This article briefly summarizes the history of that technology—digital computers, computer programming, and computer communications.

Bruce Blum wrote that “In the early 1940s ‘computer’ was a job title. A person was given a calculator and a set of formulae and then called a computer.”⁷⁴ By the late 1950s, however, when an electronic device carried out the arithmetic functions of addition, subtraction, multiplication, and division, or the logical functions of “and,” “or,” and “not,” the device was called a computer. A digital computer required a central processing unit with a primary or main memory to hold the data being processed, a program of instructions for processing the data; and circuitry to

perform arithmetic and logic operations and to control execution of instructions. Peripheral equipment included secondary or auxiliary storage devices (such as magnetic tapes and disks); data-input devices (such as keyboards, card and tape readers, and direct input from secondary storage devices); and data-output devices (such as displays, printers, and plotters), but these peripherals are not described here.

Although the computer central processor had been the basis for most early advances in informatics, it was the computer software—the computer languages, programs, procedures, and documentation—that made the hardware usable for applications. The first computer programs used the lowest level of machine codes for instructions to process data. Higher-level languages soon evolved to more efficiently program increasingly powerful computers and to better meet the requirements of different users. Programming thus evolved from first-generation machine languages to fourth-generation applications-oriented languages. By the 1980s, many of the most commonly used programs were commercially available, and most computer users did little programming themselves.

During the same four decades, computer communications moved from copper wires to fiberoptic ca-

Correspondence and reprints: Morris F. Collen, MD, Division of Research, The Permanente Medical Group, 3451 Piedmont Avenue, Oakland, CA 94611.

Received for publication: 5/20/93; accepted for publication: 11/6/93.

bles. At present, networks join computers and users with worldwide, universal, integrated digital connectivity. With respect to large medical information systems, the most important innovation in the 1980s was probably the evolution of asynchronous and synchronous connections among machines into more general local, national, and worldwide high-speed networks. Using network technology, a hospital could unite all the different computers throughout its various departments, even though the computers were located in geographically dispersed locations.

Digital Computers

Mechanical and Electromechanical Digital Computers*

In the latter part of the nineteenth century, John Shaw Billings, a physician and director of the Army Surgeon General's Library (which later became the National Library of Medicine¹³), started a chain of events that led to the development of the modern digital computer in the United States. Billings was asked to assist the Census Bureau with the 1880 and 1890 census, and he suggested to Herman Hollerith, an engineer, that "... there ought to be a machine for doing the purely mechanical work of tabulating population and similar statistics . . . . He thought of using cards with the description of the individual punched in the edge of the card."⁶² Hollerith invented, in 1882, a paper card with 288 locations for holes that were punched out by a hand-operated machine.† He went on to build machines for electrically punching and sorting these cards. The 1890 census data on 62 million people were processed in one month using 56 of Hollerith's machines. Hollerith established the Tabulating Machines Company in 1896, and in 1911 sold it to the Computing-Tabulating-Recording (CTR) Corporation. Thomas J. Watson, Sr., joined CTR in 1914, and changed its name in 1924 to the International Business Machines (IBM) Corporation. Thus, a physician, Billings, laid the conceptual foundation for the subsequent development of digital computers in the United States.

The earliest digital data processors were mechanical calculators that had been available since Pascal invented the first one in the mid-1600s. In the early 1830s, a Cambridge mathematician, Charles Babbage, built an automated mechanical computing machine

with gears and linkages, which he called a Difference Engine. In the mid-1830s, Babbage conceived, but never completed, a more advanced Analytical Engine that would have become the first mechanical digital computer.¹¹⁵ In 1940, George R. Stibitz and his colleagues at Bell Laboratories developed a partially automated digital computer that had electronically controlled mechanical (magnetic) switches called relays. The "on" and "off" positions of the relay switches represented the numbers 0 and 1 in the binary, or base 2, system.⁴⁶ Stibitz exploited the relative simplicity of using the binary system for a digital computer. Every number, letter, and punctuation symbol was represented by a unique group of bits.‡ Previous calculators and computers had processed numbers by the decimal system, which required multiples of ten gear teeth or switches in order to count.

The world's first fully functional, program-controlled, general-purpose, electromechanical digital computer was completed in 1941 by Konrad Zuse, in Germany. His machine used relay switches and was based on the binary system. Due to World War II, Zuse's work received little recognition in the United States. Howard H. Aiken, at Harvard, working with IBM engineers, built the Mark I electromechanical computer in 1943.⁶² The Mark I was a program-controlled computer that was based on the decimal system. All machine operators were performed electromechanically by wheel counters and relay switches.⁷

First-Generation Electronic Computers and the Early Computer Industry

With the advent of modern electronics, the flow of electrical current replaced the mechanical moving parts of earlier computers, and vacuum tubes replaced electromechanical switches. Credit for the invention of the first generation of all-electronic digital computers has been given generally to the mathematicians Alan M. Turing and M. H. A. Newman and their colleagues at the Bletchley Research Establishment in England.⁹⁶ Their machine, called the Colossus, was installed and working in December 1943.⁶⁰ The Colossus was used by the British during World War II to carry out the many logical steps needed to break German message codes. It showed that computers could be used for purposes other than just processing numbers.

The first electronic digital computer built in the United States was completed in 1946 by John W. Mauchly,

‡Later, the American Standard Code for Information Exchange (ASCII) assigned an eight-digit binary code to each letter, so that 01000001 was letter A, 01000010 was B, 01000011 was letter C, and so on.

*Outside this section, the term "digital computers" is used to connote "electronic digital computers."

†The punch card was designed by Hollerith to be equal to the size of a dollar bill to avoid making new equipment for storing the cards.

J. Presper Eckert, and their coworkers, at the Moore School of the University of Pennsylvania. It was called ENIAC[§] (Electronic Numerical Integrator and Calculator).⁹ ENIAC was secretly built, under contract with U.S. Army Ballistics Research Laboratory, to calculate trajectories of shells (ballistics tables) for use by gunners in World War II. A precursor to ENIAC, which some consider to have been the first electronic digital computer, had been constructed by Dr. John V. Atanasoff, a physicist at Iowa State University, between 1937 and 1942.^{91,96}

In 1945, John von Neuman, at the Princeton Institute for Advanced Study, devised a method for storing programs (i.e., storing instructions as distinguished from storing data) within a computer by using an electrically alterable memory capable of storing both the program instructions and the data to be used in calculations. Previously, a machine such as ENIAC had performed a specific sequence of calculations and its circuits had had to be rewired for any other sequence.¹⁶ Von Neuman's revolutionary design, referred to as the von Neuman machine, led to the ability to manipulate machine commands by arithmetic operations rather than by rewiring circuit boards.

Eckert and Mauchly recognized the potential usefulness of their computer in science and industry, and obtained patents. They left the University of Pennsylvania and set up the first commercial computer company in the United States, the Eckert and Mauchly Electronic Control Company of Philadelphia. In the late 1940s, they used von Neuman's stored-program technology to develop UNIVAC (Universal Automatic Computer) I, with 5,000 vacuum tubes. This computer was the first to handle both numeric and alphabetic data, although it was still based on the decimal system. In 1950, Remington Rand, founded in 1911 to make typewriters and tabulating equipment, took control of the Eckert and Mauchly Company. In 1951, Remington Rand completed the first UNIVAC for the Census Bureau, to be used in analyzing the 1950 census.

The IBM Company, which at the end of the 1930s was the country's leading business machine manufacturer, began in 1939 to finance the construction of Aiken's MARK I at the Harvard Computation Laboratory. In 1945, IBM initiated work on its first internally developed computer, the IBM 604, which was marketed in 1948. The IBM 604 was an electronic multiplier with 1,400 vacuum tubes and a plug board for wiring simple instructions. In 1952, Thomas Wat-

son, Jr., became the president of IBM. He directed the manufacturing of the 701 Defense Calculator, a binary computer with 4,000 tubes that was faster than the UNIVAC. The IBM 701 was used to support the needs of the Korean War.⁵⁶

The earliest patent for magnetic core memory was obtained in 1949 by An Wang, then at the Harvard Computation Laboratory. Wang sold his patent for magnetic core memory to IBM, and founded Wang Laboratories in 1951. Jay Forrester, at the Massachusetts Institute of Technology (MIT) Lincoln Laboratory, improved the magnetic cores by fabricating them from ferrite mixtures and stringing them on three-dimensional grids. These were first installed in 1953 in the Whirlwind computer made for the Office of Naval Research. They worked so well that magnetic core became the basic element of computer primary memory until the advent of the microchip in the late 1960s. "As the first real-time computer, Whirlwind was the source of more significant technological innovations than any other computer . . . it was the first 16-bit computer which paved the way for the development of the minicomputer in the mid-1960s."⁶²

The use of digital computers for medical purposes in the United States had been pioneered by Robert Ledley in 1950, when he conducted research in computer applications to dental projects at the National Bureau of Standards. Ledley used the Bureau's Standard Eastern Automatic Computer (SEAC), which he identified as ". . . the world's first high-speed electronic digital computer in which the programs were stored digitally in the computer's memory (. . . the UNIVAC was still not operational)."¹¹³

In 1956, IBM released the IBM 704 scientific computer with magnetic core memory, FORTRAN programming, and a cathode-ray tube monitor with some graphics capability.³⁵ The IBM 704 was one of the earliest computers used for biomedical research.

Second- and Third-Generation Large Digital Computers

The second generation of digital computers was introduced in 1958. Instead of using vacuum tubes, these computers employed transistors.[¶] Transistor-based computers were wired in much the same way as vacuum tube-based computers but they were much smaller, required less power, and were more reliable. In 1959, IBM began marketing its first transistorized computer, the 7090. It contained 32,768 words of

[§]ENIAC occupied a space 30 by 50 feet, weighed 30 tons, and used 18,000 vacuum tubes as its active arithmetic and logic elements.

[¶]William Shockley, John Bardeen, and Walter Brattain at Bell Laboratories invented transistors in 1948, and won the Nobel prize for physics for their accomplishment.

memory (with each word 36 bits in length) and employed magnetic tape for secondary storage.⁵⁶ By the end of the 1950s, IBM had three-fourths of the computer market in the United States. IBM continued through the end of the 1980s to be "... the world's most profitable industrial company."⁸¹

In 1959, Jack Kilby at Texas Instruments and Robert Noyce at Fairchild Semiconductor independently were able to make the crystal in a transistor serve as its own circuit board.⁴⁵ Kilby's device required putting the circuit components together by hand. Noyce's group fabricated the components of a circuit (the resistors, capacitors, transistors, and the interconnecting wireless conductive pathways) all on one surface of a flat wafer of silicon. They thereby created the first miniature integrated circuit on a chip.³⁰ In 1961, Fairchild and Texas Instruments introduced logic chips that, in addition to the arithmetic AND function, performed the Boolean operations OR and NOT (NOR, Not OR). "By stringing [logic] gates together in clever ways, engineers endow computers with the power to make decisions."⁶²

The invention of the computer chip with its integrated circuits produced dramatic changes in computers, and was probably the most important event to usher in the information age. In 1963, third-generation computers appeared in the form of solid-state integrated circuits that consisted of hundreds of transistors, diodes, and resistors embedded on one or more tiny silicon chips, a process called large-scale integration (LSI).⁷⁴ Once again, computers became smaller, required less power, and were more reliable. "The manufacture of computer circuits became more like printing and less like a traditional assembly process . . . . The makers worked like photoengravers with masks and light-sensitive chemicals, etching away a layer here, adjusting a new one there."⁵⁷ Beginning in the late 1960s, the fourth generation of computers exploited very-large-scale integration (VLSI) that contained thousands of components on a very tiny silicon chip. They greatly increased performance despite their smaller size and lower cost. By the early 1980s, it could be stated that, "A silicon flake a quarter inch on a side can hold a million electronic components, ten times more than the 30-ton ENIAC . . . . This prodigy is 30,000 times as cheap as ENIAC, draws the power of a night light instead of a hundred lighthouses, and in some versions performs 200 times as many [calculations a second] as ENIAC ever could."⁴⁵

Noyce left Fairchild in 1968 to start an integrated circuit design and manufacturing company called Intel. Intel introduced semiconductor, random-access

memory (RAM) chips that soon replaced magnetic core memory for primary computer memory. Memory chips contained transistors in intersecting rows and columns, from which the bytes could be retrieved individually. This permitted the construction of a hierarchy of computers of different sizes. The IBM system/360 series introduced in 1964 was a family of six of the earliest third-generation mainframe computers, all using the same operating system (OS/360) that allowed data-processing operations to grow from the smallest machine in the series to the largest without the need to rewrite essential programs. The 360 led the industry into modern commercial computing and was the basis for IBM's spectacular growth. In the early 1970s, IBM replaced its 360 series with the system/370 series made entirely of integrated circuits.

Although the first version of the NLM MEDLARS/MEDLINE system was implemented by General Electric on a Honeywell System in 1963, it was moved to an IBM 360/50 machine in 1965. It "... was replaced by a 370/155, this by a 370/158, and followed in 1977 by a twin 370/158 multiprocessor."⁴⁸ This migration demonstrates the importance of an upgrade path to keep ahead of demand. In the 1980s, MEDLARS/MEDLINE was installed on an IBM 3033 machine.

In 1965, Seymour Cray of Control Data Corporation (CDC) gave birth to the supercomputer industry by designing the CDC 6600 machine, which contained six computer processors working in parallel. The CDC 6600 "... in its time was the most powerful machine built."⁸² In the late 1970s, CDC built the Cyber 205 and Cray Research built the Cray-1, with the ability to carry out many similar operations in parallel, or on different aspects of the problem concurrently.⁴⁷

Minicomputers and Microcomputers

Whereas general purpose, third-generation computers had evolved to perform a broad range of tasks, there emerged in the 1960s special-purpose, smaller computers designed to perform with greater speed, economy, or convenience a single restricted set of tasks.⁵ In 1962, Wesley A. Clark and Charles E. Molnar at the Lincoln Laboratory of the Massachusetts Institute of Technology demonstrated a small, special-purpose Laboratory Instrumentation Computer, LINC, that increased the degree of control over the computer by the individual researcher.⁴ Bruce Waxman, executive secretary of the Computer Science Advisory Committee of the National Institutes of Health, arranged for a grant for the construction and evaluation of 12 LINC's to be distributed in 1963 to various computer science research laboratories in the

United States. Additional development of the LINC occurred at the University of Wisconsin (Jersey Rose), and at the University of Washington at St. Louis (Clark, Molnar, and Jerome Cox).¹¹⁴

In 1964, the Digital Equipment Company (DEC), founded in 1957 by Kenneth Olsen,** took over commercial production of the LINC machines.⁸⁵ The LINC was the first of a number of smaller, low-cost, minicomputers that appeared in the 1960s. C. Gordon Bell designed the first DEC Programmed Data Processor (PDP). The PDP-8 was released in 1965, and this computer "... helped coin the term minicomputer."⁸⁹ In 1966, Hewlett-Packard introduced its Model 2116A minicomputer, which was more powerful than the LINC, and by 1970, Hewlett-Packard was second only to DEC in the minicomputer market. Minicomputers were soon used for many medical applications, since they outperformed the more expensive large mainframe computers for certain input/output processing tasks, and the demands for processing large numerical calculations in clinical medicine was infrequent.¹⁴ In 1978, DEC introduced its VAX series of minicomputers, which had a more flexible and modular architecture and were faster than the PDPs.

Whereas the central processing unit of an electronic digital computer had previously been formed by combining a large number of individual components, the central processing unit of a microcomputer consisted of one or more large-scale integrated circuits on silicon chips. In 1969, Marcian E. Hoff, Jr., at Intel, fabricated the first central processing unit on a single chip. Gilbert Hyatt filed a patent application in 1970 for a prototype microprocessor using integrated circuits.¹⁰⁹ In 1971, John Blankenbaker assembled what was considered by a panel of judges to be the first personal computer; and 40 of his \$750 Kenbak machines were sold.⁷⁶

Intel developed a series of microprocessor chips that revolutionized the personal computer industry. In 1975, Intel's 8080 microprocessor was the basis for the Altair 8800, which became the next commercially available personal computer.⁷⁴ In 1980, Intel introduced the 8088 chip, which contained 2,300 transistors and performed 60 thousand operations per second (KOPS), or 0.06 millions of instructions per second (MIPS). In 1981, IBM introduced its personal computer, the IBM PC, using the Intel 8088 chip, 16 K bytes of random-access memory, the Microsoft DOS

operating system, floppy disk drives, a variety of display monitors, and a keyboard detachable from the computer.⁵³ IBM chose not to make technical information about the PC confidential, which soon resulted in the appearance of a great many "clones"—machines not manufactured by IBM, but with similar designs and components that performed identical functions. The IBM-PC was rated in 1988 by *Consumer Reports* as "... the most successful personal computer design of all time."⁹²

In 1976, Stephen Wazniak joined Steven Jobs to found the Apple Computer Company. Wazniak designed the Apple I and Apple II machines (based on the Motorola 6502 chip), which appeared on the market in 1976 and 1977, respectively. In the early 1970s, Alan Kay and associates at the Xerox Palo Alto Research Center (PARC) developed a prototype personal computer with a graphic user interface called Alto and an interactive programming language called Smalltalk. Alto had an operating system for its graphically-oriented computer with symbols, "... icons representing files, documents, and programs, and a pointing device in the form of a 'mouse' for selecting programs, documents, and complex commands from menus of choice ... [and] a CRT (cathode-ray tube) display as a window into a much larger document or database."⁷⁸ Many of the Alto features were incorporated into the Apple Macintosh, released in 1984. Apple's Macintosh was innovative, with a compact, relatively portable, vertically-designed computer with a built-in high-resolution monitor, floppy disk drives, a Motorola 68000 central processor chip, 128 K bytes of random-access memory, and its own operating system. The Macintosh operating system featured a bit-mapped graphic user interface that could support applications with multiple windows (displays within a display). The Macintosh became very popular within the educational profession because of its easy-to-learn, straightforward user interface.⁶⁷

The power of a microprocessor is greatly influenced by its scale of integration (the number of transistors on a chip), its word size (which governs the width of the computer's data path) and the frequency of its electronic clock (which synchronizes the computer's operations).⁵¹ In 1980, Intel's 8080 chip contained 2,300 transistors. In 1986, Intel's 80386 chip contained 750,000 transistors; and in 1989, Intel released the 80486 chip containing 1.2 million transistors. The Pentium chip, released by Intel in 1992, contains 3.1 million transistors. Over the same period of time, the data path of commonly available microprocessors has increased from 8 bits to 64 bits, and the clock frequency from 8 MHz to 66 MHz or more. Memory chips have increased in their storage capacity and performance in

**Olsen had worked with Forrester at MIT on the Whirlwind computer, and then with Molnar and Clark at MIT's Lincoln Laboratory.

parallel with the development of increasingly powerful central processor chips.

As processors achieved phenomenal speeds, new system architectures were developed. Early first-generation computers were "hard-wired"—their circuits were their programs. Later, during the 1960s, the central processing unit could interpret and execute instructions stored directly in primary memory: ROM (read-only memory) and RAM (random-access memory). Later-generation central processing units utilized complex-instruction-set computing (CISC).⁸⁸ Such machines "made extensive use of microprogramming, building instructions out of series of microinstructions that were stored in the ROM within the CPU."⁸⁷ Complex (CISC) instructions usually required multiple clock cycles for machine execution, introducing potential inefficiencies. An alternative to CISC computers, known as reduced-instruction-set computers (RISC), was developed by IBM (and subsequently others) by the early 1980s. The RISC omitted the complex hard-wired microprograms of the CISC, leaving only an efficient and essential set of instructions hard-wired in the CPU. The RISC design required special language compilers that translated higher-level languages into machine language code and restructured the programs to run more efficiently. Although RISC processors used more simple instructions in place of fewer complex instructions, they usually provided an improved performance through an increase in processing speed.⁹⁹

Computer Programming Languages

Definition of Computer Program

Computers can accept instructions in their native codes only (i.e., using the instruction sets with which they were designed). Machine language is the system of codes by which the instructions and the data must be represented internally. A programmer writing in a computer's low-level machine language uses machine instructions to address memory cells for storing data, accumulators to add and subtract numbers, and registers to store operands and results. Some codes represent instructions for the central processor (such as shifting the contents of a register one bit to the left or right); some codes move the data from the accumulator into the main memory; some codes represent data or information about data; and some codes point to locations (addresses) in memory.

As different generations of computers evolved, so did different generations, or levels, of programming languages.⁷³ More powerful programming languages abstracted sequences of machine-level instructions into

conceptually useful operations that could be invoked with a single mnemonic (from the Greek, meaning "to remember") command. "Programming languages . . . have been developed primarily since the late 1950s to simulate as closely as possible the natural languages people use. They employ English [or other human language] words and well-known mathematical symbols as much as possible."²⁵ Each programming language is composed of a set of statements that are based on a vocabulary of symbols. Greenes defined a programming language ". . . as a formal language used to facilitate a description, by a human user, of a procedure for solving a problem or a task. This description is a computer program, which must be translated by another program (a compiler or interpreter) into a form understandable by the computer hardware itself before it is actually executed."⁵⁹ Now languages are emerging that are declarative (instructing the machine by listing the tasks to be completed) rather than procedural (instructing the machine about how to perform the tasks by listing the discrete steps to be taken).

First- and Second-generation Programming Languages

Augusta Ada^{††} Byron, the Countess of Lovelace and the daughter of Lord Byron, is credited by many as being the first programmer of a digital computer. She collaborated with Charles Babbage in England during the 1840s to develop a variety of programs for Babbage's Analytic Engine, even though he never actually completed the engine.¹¹⁵

In 1943, Aiken used punched paper tape to enter instructions into the electromechanical Mark I computer, which was used to generate mathematical tables. In the ENIAC in 1946, the first electronic digital computer, the computer programs of instructions were wiring diagrams that showed how to set the machine's plug boards and switches to calculate the specific ballistic trajectories needed. In 1944, John von Neuman developed the concept of a stored program within the computer, and first used this historic development in 1952 in the EDVAC computer. He demonstrated that the instructions for the computer, always entered before by plugged boards or punched paper tape, could be stored in the computer's electronic memory as numbers and treated in exactly the same manner as numerical data. For the first time logical choices could be made inside the machine and the instructions could be modified by the computer.¹⁸ Computer programming, as practiced in the following decades, began at that time.

^{††}In the 1970s, a programming language was named Ada in honor of Ada Byron.

In the 1940s, to reduce the tedium of writing in machine code, programmers invented a symbolic notation called assembly language (or assembly code). Instead of writing down the binary digits for each machine instruction, the programmer wrote a short English word (such as "add" or "load") that would be translated automatically into the appropriate machine instructions. These English terms were more easily remembered than numerical instruction codes; and these came to be called mnemonic codes.³ Such second-generation low-level languages used for writing programs to translate instruction mnemonics to machine-executable binary codes were called assemblers. Assemblers were generally written to meet the design requirements of specific computers. To this day, programs that require extreme efficiency are written in assemblers, since assembly language gives the most direct and efficient control over a program's operation.

Third-generation Computer Languages

In contrast to assembly language, instructions in higher-level languages combined greater numbers of machine-level operations into single higher-level language commands. Programs written in higher-level languages tended to be shorter, quicker to write, and less prone to error, and some languages introduced the capability of being run on different computers. Since the late 1950s, most computer languages have been written in high-level languages that Ledley called automatic programming " . . . by making the computer itself help prepare the program or code, minimizing the amount of writing a programmer need do."¹

The term "compiler" was first used in 1951 by Grace Hopper,^{††} then at Remington-Rand Univac, to describe the first translator program.⁶⁹ A compiler program took the complete program written in the source code of the higher-level language and translated it into the machine-language code, which was stored as an executable file in the computer. An alternative approach to programming used an interpreter that scanned and parsed each line of the program-source code, then interpreted and changed it into machine-language code each time it was used. Thus, interpreted programs could be tested and changed as they were entered line by line, and as a result they were

easier to write and change. For an interpreted program to be executed, the interpreter had to be present in main memory and was a potential source of heavy processing overhead. In contrast, once a program had been compiled, the compiler was no longer needed, resulting in faster overall performance.

FORTRAN (FORMula TRANslator), the most influential early high-level programming language, was developed by John Backus and his associates at IBM in 1957.⁶⁹ FORTRAN provided a compiler to replace the tedious coding of assembly language. FORTRAN continued to be widely used through the 1970s and 1980s, primarily because of its special capabilities for numerical calculations, and because many useful subroutines were developed over the years in FORTRAN. APL (A Programming Language) was developed in the mid 1960s by Kenneth Iverson at Harvard for mathematical computation using vectors and matrices. Whereas FORTRAN required definitions of arrays and operated on one element in a matrix at a time, APL used symbols to execute these operations with minimal procedural definitions.^{38,75} However, standardization of FORTRAN facilitated the transfer software among machines of different vendors,⁶³ and made FORTRAN extremely popular. In 1987, it was ". . . estimated that as much as 25% of the world's available machine cycles run with code generated by some form of FORTRAN."⁸³

COBOL (COMmon Business-Oriented Language) was created in 1960 by a joint committee of computer manufacturers and users, with government and academic representatives interested in developing a high-level programming language for business data processing that would use ordinary English statements and be usable by any computer. Because of the pervasiveness of business functions, COBOL became the most widely used programming language of the 1970s and 1980s.⁸⁴ Like FORTRAN, COBOL used a compiler and was not an interactive language. From the 1960s through the 1980s, COBOL was used almost universally for the business and accounting functions in hospitals.

BASIC (Beginners All-purpose Symbolic Instruction Code) was developed in the early 1960s by John Kemeny and Thomas Kurtz at Dartmouth as a language, modeled after FORTRAN, for introductory courses in teaching computer programming. BASIC was an interactive, interpretive language, the program was converted line by line into the computer's machine language, error messages were simple for students to understand, and program changes were easy to make and test. BASIC was relatively easy to learn and to use; and it became one of the most widely

^{††}Hopper is also credited with having ". . . coined the term 'bug' after a moth stopped the Mark II she was working on dead in its tracks" [Runyan,⁸⁹ page 64]. To correct this programming "bug," Hopper used a tweezers to remove the moth from one of the relay switches in the large electromechanical computer. To find a "bug" in a complex compiled program is indeed a much more difficult problem.

used high-level languages. The earliest BASIC program for microcomputers was developed in 1975 by William Gates and Paul Allen for the Altair computer. On the basis of this product, they founded Microsoft.¹⁰⁶ By the 1980s, BASIC was available on almost every commercial personal computer.

MUMPS (Massachusetts General Hospital Utility Multi-Programming System) was developed in 1966 by Neil Pappalardo, Curtis Marble, and Robert Greenes in Octo Barnett's Laboratory of Computer Science at the Massachusetts General Hospital (MGH). MUMPS was created to implement the MGH modular hospital information system.²¹ Barnett wrote that: "The goal of the MUMPS system was to combine a simple yet powerful high-level language with an easy-to-use database handling system."²³ The first publication describing MUMPS was by Greenes and associates in 1969. They reported that MUMPS allowed "... a programming session to take the form of a conversational dialogue between the programmer and the terminal device, thus minimizing the user's time in programming."⁸ Barnett described the important characteristics of MUMPS as follows: "(1) The language is interpretive, thus facilitating rapid development of new applications; (2) the language has powerful string-manipulating commands facilitating the data management of the nonnumerical data which make up the largest part of medical information; (3) the file structure is a sparsely-filled tree-structured array where space is allocated only as needed allowing efficient use of disk storage space; (4) the method of storing data in this hierarchical storage array allows relative independence of access by the different modules (and thus relative independence of the development of modules) and yet easy exchange of data (thus facilitating the development of an integrated medical record); and (5) a relatively large number of users can be supported simultaneously."²⁰ MUMPS was probably the most commonly used programming language in the United States for clinically oriented medical applications during the 1970s and 1980s. COSTAR, a successful outpatient record system developed during the late 1960s and early 1970s, was developed in MUMPS. More recently, DXplain, a diagnostic decision-support system, was developed using MUMPS. By the end of the 1980s, both the Department of Defense and the Veterans Administration were installing nationwide MUMPS-based medical information systems.

Because early versions of MUMPS were very popular, and early MUMPS was an interpretive language,^{8,33,104} the comparative merits of compilers and interpreters were often discussed by early medical application developers. In 1986, Lewkowicz wrote,

"Most current MUMPS implementations now use a version of compiled code that offers the advantages of an interpreted language without the associated overhead required in interpreting each line of code before execution."⁸⁰ MUMPS became commercially available in 1969 when Pappalardo and Marble formed Meditech. After several years of work by its standardization committees, MUMPS was accepted as a standard language in 1977 by the American National Standards Institute (ANSI). At that time, MUMPS joined COBOL, FORTRAN, and PL/1 as ANSI Standard Languages. In 1983, it was reported that MUMPS had more than 4,500 installations around the world and also 500 or more microcomputer users.⁶¹ Excellent reviews of the history and uses of MUMPS were published by Walters, Bowie, and Wilcox⁶¹ and by Blum and Orthner.¹⁰⁴

Even though ALGOL (Algorithmic Language) was not used widely, it was the precursor of several more widely used languages. ALGOL became available in 1960 for the Burroughs B-5000 Computer.² It was a highly structured, compiled language that used English-like instructions and conventional algebra terms in its statements.⁷⁵ ALGOL used blocks of instructions in a modular fashion, and allowed assignment of variable values to be nested within other statements. ALGOL strongly influenced the development of modern programming languages such as Programming Language One (PL/1), PASCAL, C, and Ada.⁶³ As FORTRAN was somewhat limited in its text-handling capabilities, PL/1 was developed by a committee of IBM users in the early 1960s as a multipurpose language that incorporated features of FORTRAN for its scientific processing and of COBOL for its business and data-handling capabilities. Since the 1970s, PL/1 has been used to develop several hospital information systems. PASCAL, named after seventeenth-century mathematician Blaise Pascal, was developed by Niklaus Wirth in the late 1960s to teach and encourage structured programming. One of Wirth's goals was to develop a language that was independent of hardware.³⁸ Several medical decision-support programs have been written in PASCAL.^{§§} The language ADA^{¶¶} was developed in the 1970s for the Department of Defense (DoD). Ada was intended to be used as a common standard language for all DoD applications, and to be independent of any hardware.⁶⁵

GEMISCH (GEneralized Medical Information System for Community Health), was developed in 1969 by

§§Quick Medical Reference (QMR) was developed using PASCAL in 1985.

¶¶The name Ada is a registered trademark of DoD.

William Ed Hammond, William W. Stead, and M. J. Straube at Duke University, using a DEC minicomputer. GEMISCH was an interactive, high-level, multi-user database management language that supported a variety of files and provided an extensive text string-manipulation program, mathematical and Boolean logic capability, and input- and output-control programs.^{19,111} The TMR (The Medical Record) Program was developed using the GEMISCH programming language.¹⁰²

The C language was developed by Dennis Ritchie and Ken Thompson at Bell Laboratories in the mid-1970s as a structured programming language that used block structures of statements similar to ALGOL and PASCAL and produced efficient code that lent itself to systems programming. The C language was used to program the UNIX operating systems.⁹⁴ The C language became popular because a source code written in C would run on most computers. The C language, and particularly its successor C++, was used to implement object-oriented programming.^{95,103}

The third-generation languages described above were imperative or procedural languages that required the use of imperatives (or commands) in programming with consideration of computer addresses and storage schemes, and told the computer exactly how to take each step. Fourth-generation languages were more directed to functional, nonprocedural programming that allowed the programmers to concentrate more on the logical purposes of the programs, and told the computer what to do rather than how to do it.

The LISP (LIST Processing) language was developed in the late 1950s by John McCarthy at the Massachusetts Institute of Technology as a research tool to study artificial intelligence.⁵⁴ LISP was a high-level, interpreted language that used lists as its only data structure for both programs and data. A list was any group or sequence of elements, called atoms, structured in a particular order, and enclosed within parentheses.²⁹ LISP could manipulate any list of numbers, words, diseases, symptoms, symbols, deductions, and LISP statements; and lists could even be contained within lists. LISP was widely used by researchers in artificial intelligence because human thought involved more symbol manipulation than numerical calculations.³⁴ Several early medical-expert-system programs, including MYCIN and INTERNIST-I, were originally written in LISP. FORTH was developed by Charles Moore in the late 1960s, and captured many features from ALGOL and LISP.^{37,72} FORTH was generally used for specialized scientific and manufacturing applications, because its

main advantages were "The language is: compact, fast, . . . structured, . . . extensible, . . . highly portable."³⁶

Smalltalk was a language developed in the early 1970s by Alan C. Kay and associates at the Xerox Palo Alto Research Center as a programming language to control simulations on their Alto computer.^{28,41} Alto's Smalltalk became the basis for many features in the Apple Macintosh. Smalltalk had a built-in operating system with automatic storage management and was an object-oriented language ". . . in which everything is an object and objects communicate by sending messages to one another."⁵⁴ Every object was a collection of data and procedures that belonged to, and was an instance of, a class. Instructions were written to specify how instances of a class reacted when they received messages. Classes of objects could be moved from superclasses and between classes, and would inherit the attributes and variables of their instances. Smalltalk used a wide range of graphic objects from which special-purpose applications could be developed.³⁹

PROLOG (PROgramming in LOGic) was originally developed in Europe and was made commercially available in the United States in 1984.⁷⁰ Whereas earlier programs told the computer what to do by a series of explicit steps, PROLOG had pattern-matching capabilities and could be used to express logical relationships in a database, as defined by rules, such as if "A" and "B" are true, then "C" must also be true.¹⁰¹ PROLOG was used by researchers in artificial intelligence for advanced database-retrieval methods, and to process natural language. PROLOG was the initial programming language used by the Japanese for their fifth-generation knowledge-based computer systems.⁵⁸

Computer Operating Systems

Definition of Operating System

Computer operating systems control the movements of all programs and data through the computer, manage the execution of various applications programs, store programs and data, allocate shared time to multiple concurrent users, operate the computer's peripheral input-output equipment (such as keyboards, disk drives, and printers), and manage data moving from storage to telecommunications lines and peripheral devices. The software most closely associated with a particular computer is its operating system. The operating system is usually provided by the computer manufacturer, and often provides utility programs for disk maintenance and diagnostic pro-

grams to determine causes of hardware and software malfunctions.

Operating Systems for Large Computers

The first operating systems were created in the 1940s for the early electronic digital computers. They were sets of simple routines for data input and output, "... such as a program for storing binary codes from a punched paper tape into successive memory locations. The entire operating system consisted of a few hundred machine instructions."⁶⁴ Operating systems began to emerge in the 1950s. These operating systems ran in batch mode, executing in rapid succession the programs submitted by many individuals.

Operating systems soon became larger and more complex. Because of limited memory on many computers, functions of the operating system were divided. Input and output services needed by all programs were put into a kernel or core set that remained in the primary memory. System utilities were retained on disk storage and read into main memory only when needed.⁶⁴ High-level operating-systems languages were developed for specific computers or classes of computers. In the early 1960s, Burroughs used ALGOL as its system-programming language. In the late 1960s, Bell Laboratories' C language served as the basis for the UNIX operating system. The MUMPS operating system was a special-purpose software package developed to support MUMPS language applications.

In the late 1950s, a group at the Massachusetts Institute of Technology developed Project MAC, which supported the new concept of time sharing. Time-sharing operating systems switched rapidly among several user programs and gave the impression that the programs were executed simultaneously.⁴⁴ The System Development Corporation (SDC) in Santa Monica developed an early time-sharing computer system, sponsored by the Department of Defense Advanced Research Projects Agency (DARPA), that could accommodate 53 users simultaneously.⁶ By the late 1960s, time-sharing systems were used on a number of commercial machines; in the 1970s, their use became ubiquitous, constituting the default method for interacting with large, centralized machines.

Operating Systems for Minicomputers and Microcomputers

The UNIX operating system was created in 1969 by Ken Thompson, Dennis Ritchie, and Doug McIlroy at AT&T Bell Laboratories. UNIX was attractive because it supported a number of powerful system-

utility programs that could be chained together in arbitrary useful sequences. Its features included time-sharing, multi-user access (it could serve more than one user at the same time), multitasking (it could run several applications at the same time), and an open architecture (usable by different vendors' computers). UNIX became popular in the university setting because antitrust laws initially led AT&T to license the product for a nominal low fee. By 1983, 80% of colleges granting computer science degrees had adopted UNIX, and several versions of UNIX had been developed.⁸⁵ In 1984, after AT&T's breakup, it was allowed by the courts to enter the computer business and sell the UNIX operating system. Since UNIX could run on a large number of different computers, singly or in a network, it became a major competitor for the powerful operating systems needed for networks of desktop computers, workstations, and mainframe computers.

Microcomputer operating systems were both simpler and more specialized than the operating systems of mainframes and minicomputers. In 1974, the Control Program for Microcomputers (CP/M) was developed by Gary Kildall, the founder of Digital Research, for S-100, 8080-based computers, and subsequently for the IBM PC series of personal computers. By the end of the 1970s, CP/M was used worldwide in almost 200,000 installations.⁴⁰ In the early 1980s, IBM chose Microsoft's Disk Operating System (MS-DOS) for its new 8088-based PCs. MS-DOS became the most widely used operating system in the nation during the 1980s. In 1984, the Apple Macintosh, with an operating system that included a graphic user interface, was introduced. The Macintosh operating system displayed menus (lists of options available for selection) and icons (symbols representing options), from which the user could select items by clicking a mouse pointer. By the end of the 1980s, the leading microcomputer operating systems included Microsoft DOS, IBM OS/2, the Macintosh operating system, and several variations of AT&T UNIX.

Databases and Other Large-scale Application Programs

Through the decades, the complexity of applications increased. More commercial software packages came to market, and software costs became a greater proportion of total computing costs. Users found it increasingly more economical to obtain commercial application programs that were packaged to perform specific, commonly needed functions than to develop such programs directly themselves. Specialized commercial application programs included databases, word-processing programs, and spreadsheets for tab-

ular processing of numerical data. It became less necessary for the users of such software to know how to program a computer themselves.

The development of efficient computer-stored databases was essential for many medical computing applications. "The main contribution of database management science was separating questions about what information should be stored in a system from how it should be stored."⁹⁰ Gio Wiederhold of Stanford University made substantive contributions to the early design and development of medical databases.^{42,43} He defined a database as a "... collection of related data, organized so that usable data may be extracted."⁵² He defined a database-management system as the hardware and software that controlled, stored, processed, and retrieved the data in the database. Distributed database-management systems involved the management of physically dispersed data in two or more databases located in different computers, and used some means of communications for the exchange of data.

Early databases organized data using a network model, which defined data sets linked by pointers, and procedures for manipulating both. These early efforts led to a Conference on Data Systems Languages (CODASYL) in 1971, which advocated a variant of the network model, a hierarchical, treelike, branching structured database.²⁴ In a hierarchical database, connections between files, or between fields within files, were defined at the start of the database.

In the early 1970s, E. F. Codd^{10,12} at IBM in San Jose, California, developed the relational database model, which expressed all stored data in the form of two-dimensional tables. Every data element was logically accessible through the use of the table and column name, and data transformations resulted from following logical rules. The relational database model, because of its simplicity and power, became dominant. In the 1970s and 1980s, IBM and others developed a fourth-generation language to serve as a standard interface for constructing, managing, and querying relational databases—Structured Query Language (SQL).¹¹²

Beginning in the late 1980s, some database applications were programmed in object-oriented languages. Object-oriented programs treated the database as a modular collection of component items called objects, which belonged to types or classes with defined characteristics or attributes. Whereas traditional database programs separately represented data and the procedures for manipulation of the data, in an object-oriented system, the objects encapsulated both. Objects were sufficiently modular and independent that

they could be copied easily into other programs. An object-oriented database could serve a network of workstations wherein one or more computers were designated object servers that would supply applications programs with objects as needed. The result was "... to minimize workstation and server communications."¹⁰⁵

Computer Communications

Data-transmission Techniques

Early computer hardware and software had to meet specific communication requirements, or protocols, for each remote system to be connected. For serial communication over an asynchronous line (i.e., one wire had to carry all the data bits), these requirements included transmission speed, the width of each datum, the start and stop signals used to control the flow of data between computers, data-transfer methods, and error-checking rules used for transmitting the data. The most widespread standard used in the United States for serial data transfer was a binary, eight-bit code designed to represent all characters used in information exchange. This code was established in 1967 as the American Standard Code for Information Interchange (ASCII).

The wiring and switching methods originally used by public telephone networks were optimized for voice transmission, and limited the transmission speed of data to a range between 300 and 9,600 bits per second (baud). Communicating with another computer over telephone lines required a modem (modulator-demodulator) to modulate the electronic pulses representing digital data into analog signals impressed on the telephone carrier signal. To receive the data, the analog-audible signals needed to be demodulated back into the digital-pulse form.^{***} Modems varied in their speeds to transfer data. The number of times each second the line changed the signal was referred to as its modulation or baud rate. The telephone system transmitted voice and data over telephone lines in a continuous stream known as circuit switching. With this technique, once a transmission started, the circuit could be broken only after all the data had been transferred. This was not well suited to many scientific applications, where bursts of data were often interspersed with periods of silence. For example, a scientist might transmit a batch of data characterizing a problem, then wait for the remote machine to return the answer. These periods

***The first modem transfer protocol program, used by sending and receiving computers to transfer data files over a serial line, was developed in 1977 by Ward Christensen [Meeks¹¹⁰].

of line idleness were costly and prevented others from using the circuits during the interim.

In 1948, Claude Shannon introduced the concept of transmitting messages in the form of closely spaced pulses, equivalent to digital bits. Shannon's theory resulted in the development of pulse-code modulation, and this led to pulsed or digitized signals.¹⁵ In 1964, Paul Baran at the Rand Corporation, using pulse-code modulation, equipped each node, or junction, of a network with a small, high-speed computer. The computers nearest to the participants in a conversation divided the outgoing message into small segments of bits (which included control elements), or packets, then transmitted the packets as separate communications by way of the intervening communication computers to the receiving computer. The receiving computer reassembled the message packets and reconstructed the message. This process for communicating data was called packet switching and was eventually adopted by major wide-area networks such as ARPANET, Telenet, and Tymnet.²²

The ability of networks to move data is dependent on their bandwidth. Bandwidth measures the difference between the lowest and highest signal frequencies the network can transmit. Local area networks (LANs) transmit data by either baseband or broadband transmission. In baseband transmission, data are directly encoded as digital signals in a serial stream of formatted data packets on the network cable. Only one transmission signal at a time can be sent. Thus, a baseband node can either send or receive, but cannot do both simultaneously. Broadband transmission allows multiple data signals to be present on the network line at the same time. In broadband transmission, the data packets are transmitted as analog signals. The analog signals require high-speed circuitry for analog-to-digital conversion (and vice versa). Broadband networks are more expensive, but they can support many more users than baseband networks.

Network Transport Media

The earliest means for communicating data between computers were by solid wire lines or cables. For computers located at different distant sites, the earliest form of transmission was by a pair of twisted copper wires, the traditional medium for telephone lines. These lines had a narrow bandwidth that was adequate for transmitting voice signals. Pairs of wires were twisted together to decrease interference between adjacent pairs of wires. Later, for higher-speed wideband transmission, coaxial cables were used in which a single copper wire was covered with a plastic

insulation sheath, then with a wire mesh or external metal wrapping for electromagnetic shielding.

In the early 1950s, optic fibers were invented. These were glass fibers no thicker than a strand of human hair. In 1970, Corning Glass developed an optical fiber that could conduct light for at least one kilometer.³² Light signals traveled through the core of the fiber and were reflected back and forth from the core's boundary with the cladding. The fiber thus guided light rays along its length.¹⁷ AT&T pioneered further development of optic fiber technology at its Bell Laboratories. However, telephone companies did not begin to offer services with fiberoptic lines until 1983. Analog voice or data signals were converted by an optic transmitter to pulses of light from a laser or a light-emitting diode flashing on and off at very high speed. Amplifiers and repeaters were required in early fiberoptic systems to convert the attenuated light pulses to electronic signals, give them a boost of electric power, convert them back to amplified light pulses, and send them on through the next section of cable. Fiberoptic lines were immune to electrical or magnetic interference and they provided cleaner, quicker, and more reliable transmission of data, voice, and images. The higher frequency of light carried more information than could analog sound signals, so one single strand of glass fiber could carry 24,000 telephone conversations simultaneously, as compared with 4,000 conversations per pair of coaxial cables, or 24 for a pair of copper wires. By the late 1980s, fiberoptic technology had improved to the point that images could travel hundreds of kilometers without amplifiers or repeaters, using fibers transparent to conducting infrared light.⁹³ By the end of the 1980s, fiberoptic cables were rapidly replacing copper coaxial cables in telephone and communication services. This development spurred the creation of networks of integrated data systems.

For remote locations, earth-orbiting communication satellites provided another technology alternative by the 1980s. The first successful active communications satellite, Telstar, was launched in 1962, and it carried live television signals across the ocean. The first commercial satellite, Intelsat I, was launched in 1965.³¹ By the end of the 1970s, the fifth generation of the Intelsat satellites had been launched.²⁶ In the early 1980s, some of the remote networks included provision for relaying signals by such geostationary earth satellites.

Computer Networks

Computer networking evolved in the 1960s for time sharing. Time sharing provided an early means for

different users with different problems to have interactive access to a mainframe computer without having to purchase such an expensive machine. In the 1970s, minicomputers provided local data-processing capabilities at various work sites and created the need for a distributed data-processing system. Three alternative networking modes were developed to allow multiple users to have access to common computing facilities: 1) direct tielines connecting distributed computers or terminals to a central host computer; 2) modems connecting users' terminals or computers by telephone lines over a wide geographic range; and 3) local-area-network (LAN) cables connecting computers to one another.⁶⁶ A LAN typically connected on-site computers and peripheral devices, integrated departmental computing facilities, and connected local machines to a supporting computer center. By 1971, prototype LANs of distributed minicomputers were being proposed in a hospital environment.¹¹ Distributed computer systems limited the effect of the failure of one computer on the operation of the other computers.

A variety of network topologies or configurations evolved to provide links between the computers. In a star network, individual connections to a computer module radiated out from the host computer like spokes from a central hub. All communications on the network passed through the host computer. However, if the hub failed, then all communication stopped. In a ring network, all computers were connected in a loop. Each computer was connected to two others. Each message passed around the circle in a single direction, was received by the appropriate module, and was removed when it returned to the original sender. In a bus (or linear) network, all computers were connected via "drop offs" from a two-way main line. A message sent by any one computer traveled in both directions and could be received by all other computers. Bus LANs were the most common due to their lower cost, easy connectivity, and expandability. They readily permitted broadcasting from one computer to all other computers, and they were resistant to single computer failures.

In 1973, Robert Metcalfe and his group at Xerox Palo Alto Research Center developed Ethernet, which became one of the earliest baseband coaxial cable networks for high-speed communications. By 1974, many commercial networks were available, including AT&T's ISN, Control Data's CDCNet, Corvis Systems' Omninet, Datapoint's ARCnet, Digital Equipment's DECnet, IBM's SNA, and Wang's WangNet. By the end of the 1980s, there were more than 75 LAN suppliers in the United States, and a 1988 survey

reported that 30% of PC users were connected to LANs.¹⁰⁷

In the 1980s, the controlling communication software functions were moved from participating network nodes into the network itself, which then contained its own computers called network interface units. Thus, any computer in the network had to communicate only with its network interface node, which usually required a less complicated protocol and supporting software to be present on each participating node. Communications on a LAN were usually accomplished through one of two general methods. In contention or collision-detection access, each computer using the network had two connections, one for receiving data from and one for sending data to the network. Each node could send data only when no other message was being transmitted. So each computer had to listen for any traffic on the channel before sending. This method was used, for example, by Xerox's Ethernet.⁸⁶ Token-passing access used an electronic signal of a packet of bits in a special pattern (token) that was passed from computer to computer in a ring or bus network. If a computer node had no message to transmit, it passed on the token. To transmit a message, a computer node retrieved a free token, held it while communicating, and sent the message specifying the destination address, and other computers checked the message as it passed by. IBM's PCNet and Datapoint's ARCnet used the token-passing method.^{66,68}

Database integration on a LAN meant that the user who received a response to a query was unaware of where or how the data were stored. This technology involved either interfacing or integrating distributed heterogeneous data. The interfacing approach permitted exchange of data by building cross-reference tables between each pair of physical computer files in two databases to be interfaced, which transformed or mapped one physical structure into the other as required; and it allowed no technologic controls over data integrity. Executive functions for interfacing could be defined in master data dictionaries used by all interfaced databases. The integrating approach permitted actual merging of data by employing a conceptual schema that was mapped to database structures as well as to user requirements. The schema standardized data by defining a consistent set of rules to be used for all data, integrated data by pulling together the data from multiple databases, assured data quality by enforcing data-integrity constraints, and maintained data independence by isolating data-structure changes from users' programs. A master data dictionary defined, described, and managed the schema and the transforms of the users' data.⁷¹

Although it was important for networks to be concerned about the wires or cables that connected computers, it became even more important how the computers communicated data with each other—that is, the rules or protocols they used governing the format, response, time, message length, and error handling of messages. Orthner pointed out that the data-processing industry had minimized the use of the analog voice channels of the telephone industry, and generally preferred to install coaxial cables, with the result that there were myriad local area networks, each with its own proprietary communication protocol.⁹⁸ Network protocols required standardization of a variety of processes involved in communications. The International Standards Organization (ISO) fostered the development of the Open Systems Interconnection (OSI) Reference Model.

The ISO/OSI communications model used seven layers, each of which performed a well-defined function: 1) The physical layer activated and deactivated the electrical and mechanical transmission channels to various equipment. 2) The data-link layer provided for transfer of blocks of data between data-terminal equipment connected by a physical link, including sequencing, flow control, and error detection. 3) The network layer provided routing and switching of messages or packets between adjacent nodes in the network. 4) The transport layer provided an end-to-end transparent transmission channel. 5) The session layer opened communications, established a dialogue, and maintained the connection, including control and synchronization for the transfer of messages between two computers. 6) The presentation layer ensured that the message was transferred in a form the receiving computer could interpret. 7) The application layer, the only part of the system apparent to the user, provided services that facilitated information exchange between application processes on different computers.⁵⁵ The OSI was the starting point for Health Level Seven (HL7) interface standards for transmitting data between applications in hospital information systems.

Thus, each of the seven ISO/OSI layers had a defined set of functions and a layer protocol that established the rules for exchange with the corresponding layer in another computer. Vendors now began to consider that open systems, in the narrowest sense, meant strict adherence to the OSI standards. To permit the connection and integration of LANs with other LANs required the development of: bridges that operated at layer two of the OSI seven-layer architecture and connected one LAN to another; routers that operated at layer three and routed packets of data between dissimilar networks; and gateways that operated at

level seven and provided high-speed communications from a mainframe computer to a network. This ISO/OSI model had a great influence on the further development of large complex LANs. Ethernet supported the first three levels of the ISO model to the level of the packet, and provided system software to perform the remaining four levels of converting to and from packets logical messages. IBM's System Network Architecture (SNA) supported all seven layers of the ISO model.⁷⁴

Wide-area networks (WANs) were usually long-distance data communication networks using public telephone lines. The United States took the lead in the development of wide-area computer networks for academic research. In 1966, the Department of Defense's Advanced Research Projects Agency (DARPA) contracted with Bolt, Beranek & Newman (BB&N) to create ARPANET to link computers of different types in various colleges and institutions engaged in research for the Defense Department.⁵⁰ ARPANET began to operate in 1969 and was the world's first interactive, computer-to-computer, packet-switching network.⁸⁶ ARPANET became a national coast-to-coast network, which permitted researchers at the University of Illinois, for example, to do serious computing on computers located in San Diego. In 1972, BB&N initiated the Telenet Communications Corporation, a commercial packet-switching network that later became a subsidiary of GTE.¹⁰⁰ In the 1980s, GTE's Telenet served "... more than half a million terminal users in and around 400 U.S. cities . . . . With the advent of packet-switching [networks], the boundary between computing and communicating, once sharply etched, became blurred."⁷⁷

Tymnet, a competitor of Telenet, had built a network with more capacity than its time-sharing business could fill. To use some of the excess, Tymnet contracted with the National Institutes of Health to provide communication links between terminals installed in medical schools and a computer and bibliographic database belonging to the National Library of Medicine. "This arrangement worked out so well that other host computers were added to the network, which employed a variant of the packet-switching technology pioneered by ARPANET."⁷⁷ ARPANET was so successful that it was gradually expanded, and in the 1980s it was the major component of a supernet called Internet. Internet also included the Computer Science Network (CSNET), which was created to let academic computer scientists share computational resources and exchange mail, and the National Science Foundation Network (NSFnet) that connected 60 major research universities in the United States.⁷⁹

By the mid-1980s, communication specialists were planning a single, unified, Integrated Services Digital Network (ISDN) capable of tying together every computer and organization in the world. The goal was to provide a group of standards for worldwide, universal digital interconnectivity regardless of modality (i.e., for data, voice, and video) over ordinary telephone wires, to be potentially available from every telephone outlet.¹⁰⁸ ISDN allowed computers and telephones to communicate directly with each other and eliminated the need for modems. By the end of the 1980s, patchworks of ISDN in the United States showed that ISDN could successfully link existing equipment from multiple vendors.

In 1979, (then U.S. Senator) Albert Gore first proposed a network of information superhighways linking the nation by fiberoptic cables. Some broadband fiberoptic networks began to be introduced into large medical centers by the addition of video channels to standard baseband LAN technology. This permitted linking text, voice, and video; communicating data to and from minicomputers and laser printers; and communicating images to and from image scanners, digitized x-ray equipment, television monitors, and facsimile machines.

Summary

Over the past four decades, we have witnessed the transition from the end of the Industrial Age to the beginning of the Information Age. Computers have evolved from expensive, slow, electromechanical devices to inexpensive, miniaturized, amazingly fast components that are found pervasively throughout our society. The invention of the computer chip with its integrated circuits produced dramatic changes in computers, and was probably the most important event to usher in the information age.

Programming languages have evolved almost as dramatically as computer hardware. The first computer programs used the lowest level of machine codes for instructions to process data. Higher-level languages evolved to more efficiently program increasingly powerful computers and to better meet the requirements of different users.

Communication of data between computers in different locations was at first entirely by telephone lines, later by coaxial cables, and in the 1980s increasingly by fiberoptic lines. In the early 1980s, computing and communication technology were converging; and by the end of the 1980s, open network architecture with interconnection protocols for open systems provided compatibility of computers from

different vendors. This became the basis for further network evolution with multimedia communications.²⁷

Morris summarized the significance to hospitals of the communication advances in the 1980s as follows: "With the ability to bridge between different vendors' equipment, hospitals will be free to approach software selection on a departmental basis and look for the system that best meets their requirements."⁹⁷ Naisbitt⁴⁹ forecasted that the combined communication technologies of the telephone, computer, and television would merge into an integrated, multimedia information and communication system that could transmit all forms of data and provide rapid interactions between persons, and between persons and computers.

We have only begun to realize the effects that these profound changes in technology will have on the biomedical community. If the recent history of computers, software, and communications repeats itself, we will be living in interesting times for generations to come.

Supported in part by a contract with the National Library of Medicine (NLM). Excerpted, with permission of the Director of NLM, from portions of a book in preparation entitled: A History of Medical Informatics.

References ■

(Sorted chronologically by year and alphabetically by first author within each year)

1961

1. Ledley RS. Introduction to digital computers and automatic programming. IRE Trans Bio-Med Electronics. 1961 (July);BME-8:158-67.
2. Taylor W, Turner L, Waychoff R. A syntactical chart of ALGOL 60. Commun Assoc Comput Mach. 1961 (Sept):4.

1963

3. IBM General Information: Programming Systems Concepts. White Plains, NY:IBM Corporation, 1963.

1964

4. Clark WA, Molnar CE. The LINC: a description of the laboratory instrument computer. Ann NY Acad Sci. 1964;115:653-68.

1965

5. Cox JR. Special purpose digital computers in biology. In: Stacy RW, Waxman BD (eds). Computers in Biomedical Research. New York: Academic Press, 1965:67-99.

1966

6. Myer EP. A time-sharing multi-unit computing system. Ann NY Acad Sci. 1966;128:738-45.

1968

7. Crawford FR. Introduction to Data Processing. Englewood Cliffs, NJ: Prentice-Hall, 1968.

1969

8. Greenes RA, Papillardo AN, Marble CW, Barnett GO. Design and implementation of a clinical data management system. *Comput Biomed Res.* 1969;2:469-85.
9. Rosen S. Electronic computers: A historical survey. *Comput Surveys.* 1969 (Mar);1:7-36.

1970

10. Codd EF. A relational model of data for large shared data banks. *Comm ACM.* 1970;13(6):377-87.

1971

11. Blois MS, Henley RR. Strategies in the planning of hospital information systems. *Journee D'Informatique Medicale.* 1971:89-98.

1972

12. Codd EF. Further normalization of the data base relational model. In Rustin R (ed). *Data Base Systems.* Englewood Cliffs, NJ: Prentice-Hall, 1972:33-64.
13. Goldstine HH. The Computer from Pascal to von Neumann. Princeton, NJ: Princeton University Press, 1972:66-71.
14. Hammond WE, Lloyd SC. The role and potential of mini-computers. In Haga E, Brennan RD, et al. (eds). *Computer Techniques in Biomedicine and Medicine.* Philadelphia: Auerbach, 1972:332-44.
15. Pierce JR. Communication. *Sci. Am.* 1972;227:31-41.

1973

16. Brazier MAB. From calculating machines to computers and their adoption by the medical sciences. *Med Hist.* 1973;17:235-43.
17. Cook JS. Communication by optical fiber. *Sci Am.* 1973;229:28-35.
18. Eamos C. A Computer Perspective. Cambridge, MA: Harvard University Press, 1973.
19. Hammond WE, Brantley BA, Feagin SJ, et al. GEMISCH—a minicomputer information support system. *Proc IEEE.* 1973;61:1575-83.

1974

20. Barnett GO. Massachusetts General Hospital computer system (Boston). In Collen MF (ed). *Hospital Computer Systems.* New York: John Wiley & Sons, 1974:517-45.
21. Barnett GO. The modular hospital information system. In Stacy RW, Waxman BD (eds). *Computers in Biomedical Research.* 1974:243-85.

1975

22. Kimbleton SR, Schneider GM. Computer communication networks: approaches, objectives, and performance considerations. *Comput Surv.* 1975;7:129-66.

1976

23. Barnett GO, Souder DE, Bowie J, Justice N. MUMPS: a support for medical information systems. *Med Inform.* 1976;1:183-9.
24. Taylor RW, Frank RL. CODASYL data base management systems. *Comput Surv.* 1976;8:67-103.

1977

25. Davis RM. Evolution of computers and computing. *Science.* 1977;195:1096-102.
26. Edelson BI, Pollack L. Satellite communications. *Science.* 1977;195:1125-33.

27. Farber D, Baran P. The convergence of computing and telecommunications systems. *Science.* 1977;7:295-9.
28. Kay AC. Microelectronics and the personal computer. *Sci Am.* 1977;237:230-44.
29. McGath G. A look at LISP. *Byte.* 1977;2:156-61.
30. Noyce RN. Microelectronics. *Sci Am.* 1977;237:62-9.
31. Pierce JR. Electronics: past, present, and future. *Science.* 1977;195:1092-4.

1979

32. Boraiko AA. Fiber optics: harnessing light by a thread. *Nat Geographic.* 1979;156:516-34.
33. Bowie J. Methods of implementation of the MUMPS global database. *Med Inform.* 1979;3:151-64.
34. Kornfeld WA. Pattern-directed invocation languages. *Byte.* 1979;4:34-48.
35. Reid-Green KS. History of computers: the IBM 704. *Byte.* 1979;4:190-2.

1980

36. James JS. What is FORTH? A tutorial introduction. *Byte.* 1980;5:100-34.
37. Moore CH. The evolution of FORTH, an unusual language. *Byte.* 1980;5:76-92.
38. Pournelle J. BASIC, computer languages, and computer adventures. *Byte.* 1980;5:222-32.

1981

39. Ingalls DHH. Design principles behind Smalltalk. *Byte.* 1981;6:286-98.
40. Kildall G. CP/M: a family of 8- and 16-bit operating systems. *Byte.* 1981;6:216-32.
41. Tesler L. The Smalltalk environment. *Byte.* 1981;6:90-147.
42. Wiederhold G. Databases for health care. In Lindberg DAB, Reichertz PL (eds). *Lecture Notes in Medical Informatics.* New York: Springer-Verlag, 1981:1-75.
43. Wiederhold G. Database technology for healthcare. *J Med Syst.* 1981;5:175-96.

1982

44. Bacon G. Software. *Science.* 1982;215:775-83.
45. Boraiko AA. The chip: electronic marvel that is changing your life. *National Geographic.* 1982;162:421-456.
46. Bronzino JD. *Computer Applications for Patient Care.* Menlo Park, CA: Addison-Wesley, 1982.
47. Levine RD. Supercomputers. *Sci Am.* 1982;246:118-35.
48. Miles WD. A History of the National Library of Medicine. Washington, DC: US Government Printing Office, 1982:387.
49. Naisbitt J. *Megatrends.* New York: Warner Books, 1982.
50. Newell A, Sproul RF. Computer networks: prospects for scientists. *Science.* 1982;215:843-51.
51. Toong HD, Gupta A. Personal computers. *Sci Am.* 1982;247:86-107.
52. Wiederhold G. Databases for ambulatory care. *Proc AMIA.* 1982:79-85.
53. Williams G. A closer look at the IBM personal computer. *Byte.* 1982;7:36-66.

1983

54. Amsterdam J. Computer languages of the future. *Popular Comput.* 1983 (Sept):136-51.
55. Blaine GJ. Networks and distributed systems: a primer. *Proc MEDINFO.* 1983:1118-21.
56. Blum BI. Mainframe, minis, and micros: past, present and future. *MEDCOMP.* 1983;1:40-8.

57. Davis M. The chip at 35. *Personal Comput.* 1983 (July):127-31.
58. Feigenbaum EA, McCorduck P. *The Fifth Generation Artificial Intelligence and Japan's Computer Challenge to the World.* Reading, MA: Addison Wesley, 1983.
59. Greenes RA. Medical computing in the 1980s; operating systems and programming language issues. *J Med Syst.* 1983;7:295-9.
60. Hodges A. *Alan Turing: The Enigma.* New York: Simon and Schuster, 1983.
61. Walters RF, Bowie J, Wilcox. *MUMPS Primer.* College Park, MD: MUMPS Users Group, 1983.

1984

62. Augarten S. *Bit by Bit: An Illustrated History of Computers.* New York: Ticknor & Fields, 1984.
63. Blum BI. Programming languages. *MD Comput.* 1984;1:60-8.
64. Denning PJ, Brown RL. Operating Systems. *Sci Am.* 1984;251:94-130.
65. Edge RA, Marciniak TA. The ADA environment. *Proc SCAMC.* 1984;882-5.
66. Haney JP. Introduction to local area networks for microcomputers—characteristics, costs, implementation considerations. *Proc SCAMC.* 1984;779-85.
67. Miller MJ. Apple's Macintosh. *Popular Comput.* 1984 (Mar):80-100.
68. Simborg DW. Local area networks: why? what? what if? *MD Comput.* 1984;1:10-20.
69. Tesler LG. Programming languages. *Sci Am.* 1984;251:70-8.
70. Weiner JL. The logical record keeper: PROLOG on the IBM. *Byte.* 1984;9:125-31.

1985

71. Appleton DS. The technology of data integration. *Datamation.* 1985;32:106-16.
72. Pearson WR. Programming languages III. *MD Comput.* 1985;2:45-9.
73. Rapp W. Fourth generation languages. *Comput Health Care.* 1985 (Nov):38-40.

1986

74. Blum BI. *Clinical Information Systems.* New York: Springer-Verlag, 1986.
75. Blum BI. Programming languages. In Blum BI. *Clinical Information Systems.* New York: Springer-Verlag, 1986:112-49.
76. Bulkeley WM. Who built the first PC? *Wall Street Journal,* May 14, 1986:31.
77. Conlan R. *Understanding computers, communications.* Chicago: Time-Life Books, 1986.
78. Crecine JP. The next generation of personal computers. *Science.* 1986;231:935-43.
79. Jennings DM, Landweber LH, Fuchs IH, et al. Computer networking for scientists. *Science.* 1986;231:943-50.
80. Lewkowicz J, Walters RF. Design of an advanced MUMPS programming environment. *Proc SCAMC.* 1986:336-43.
81. *Wall Street Journal,* April 7, 1986:19.

1987

82. *Datamation.* 1987;34:66.
83. Davis SG. Fortran at 30: formula for success. *Datamation.* 1987 (April 1):47-56.
84. Gullo K. Steady as she goes. *Datamation.* 1987 (Jan 15):34:37-40.
85. Hassig L. *Understanding Computers, Revolution in Science.* Richmond, VA: Time-Life Books, 1987:59-67.
86. Kahn RF. Networks for advanced computing. *Sci Am.* 1987;257:136-43.

87. Robinson P. How much of a RISC? *Byte.* 1987;12:143-60.
88. Robinson P. Overview of programmable hardware. *Byte.* 1987;12:197-203.
89. Runyan L. The Datamation hall of fame. *Datamation.* 1987;34:56-74.
90. Walters RF. File structures for database management systems. *MD Comput.* 1987;5:30-41.

1988

91. Burks AR, Burks AW. *The First Electronic Computer: The Atanasoff Story.* Ann Arbor, MI: University of Michigan Press, 1988:38.
92. *Consumer Reports,* 1989 Buying Guide Issue. IBM clones. Mount Vernon, NY: Consumers Union of U.S., 1988;53:308-12.
93. Drexhage MG, Moynihan CT. Infrared optical fibers. *Sci Am.* 1988;259:110-6.
94. Kernighan BW, Ritchie DM. The state of C. *Byte.* 1988;13:205-10.
95. Linowes JS. It's an attitude. *Byte.* 1988;13:219-24.
96. Mackintosh AR. Dr. Atanasoff's computer. *Sci Am.* 1988;259:90-6.
97. Morris DC. Networks: the key to the future. *Comput Health Care.* 1988;9:59-62.
98. Orthner HF. New communication technologies for hospital information systems. In Bakker AR, Ball MJ, Scherrer JR, Willems JL (eds). *Towards New Hospital Information Systems.* Amsterdam, The Netherlands: North Holland, 1988:203-12.
99. Rosch WL. Calculated RISC. *PC Comput.* 1988;1:172-88.
100. Schultz B. The evolution of ARPANET. *Datamation.* 1988;35:71-4.
101. Shifman M, Jelovsek FR. Prolog: a language for programming medical logic. *MD Comput.* 1988;5:36-40.
102. Stead WW, Hammond WE. Computer-based medical records: the centerpiece of TMR. *MD Comput.* 1988;5(5):48-62.
103. Stroustrup B. A better C? *Byte.* 1988;13:215-16D.

1989

104. Blum BI, Orthner HF. The MUMPS programming language. In Helmuth HF, Blum BI (eds). *Implementing Health Care Information Systems.* New York: Springer-Verlag, 1989:396-420.
105. Dawson J. A family of models. *Byte.* 1989;14:277-82.
106. Gates B. The 25th birthday of Basic. *Byte.* 1989;14:268-76.
107. Hodges P. LAN growth surges. *Datamation.* 1989;36:32-6.
108. Johnson B. The cult of ISDN. *PC Comput.* 1989;2:144-51.
109. Leibowitz MR. Founding father: Robert Noyce. *PC Comput.* 1989;2:95-100.
110. Meeks BN. Dialing up 1990. *Byte.* 1989;14:273-8.
111. Straube MJ, Hammond WE, Stead WW. The GEMISCH programming language. In Orthner HF, Blum BI (eds). *Implementing Health Care Information Systems.* New York: Springer-Verlag, 1989:384-95.
112. VanName ML, Catchings B. SQL: a database language sequel to dBASE. *Byte.* 1989;14:175-82.

1990

113. Ledley RS. Medical informatics: a personal view of sowing the seeds. In Blum BI, Duncan K (eds). *A History of Medical Informatics.* New York: Addison-Wesley, 1990:84-110.
114. Molnar CE, Clark WA. Development of the LINC. In Blum BI, Duncan K (eds). *A History of Medical Informatics.* New York: Addison-Wesley, 1990:119-40.

1992

115. Toole BA. *Ada, the Enchantress of Numbers.* Mill Valley, CA: Strawberry Press, 1992.