human mind can make an important contribution to that progress. It is a belief of this kind that persuades researchers in artificial intelligence that their endeavor is an important and exciting chapter in man's great intellectual adventure.

### Summary

From an economic standpoint, the modern computer is simply the most recent of a long line of new technologies that increase productivity and cause a gradual shift from manufacturing to service employment. The empirical evidence provides no support for the claim sometimes made that the computer "mechanizes" and "dehumanizes" work. Perhaps the greatest significance of the computer lies in its impact on Man's view of himself. No longer accepting the geocentric view of the universe, he now begins to learn that mind, too, is a phenomenon of nature, explainable in terms of simple mechanisms. Thus the computer aids him to obey, for the first time, the ancient injunction, "Know thyself."

### References and Notes

1. A few years ago, Newell and I erred in predicting that certain specific developments in artificial intelligence were going to take place "within 10 years." The fact that we were optimistic about the time scale has blinded a number of critics to the basic soundness of our characterization of the nature and directions of artificial intelligence. I shall try not to make the same mistake here of predicting that very specific things will occur at very specific times. See H. A. Simon and A. Newell, *Oper. Res.* **6**, 1 (1958).
2. For more detailed discussions of these topics, see H. A. Simon, *The New Science of Management Decision* (Prentice-Hall, Englewood Cliffs, N.J., ed. 3, 1977).
3. The position that computers can be programmed to simulate an indefinite range of human thinking processes is developed in detail, and a large body of supporting evidence is examined in A. Newell and H. A. Simon, *Human Problem Solving* (Prentice-Hall, Englewood Cliffs, N.J., 1972); and in J. R. Anderson and G. H. Bower, *Human Associative Memory* (Winston, Washington, D.C., 1973).
4. For a fuller discussion of this evidence, see (*2*, chap. 4).
5. K. von Clausewitz, *On War* (Modern Library, New York, 1943), p. 596.
6. The polls under discussion were conducted by the Survey Research Centers of the Universities of Michigan and California, the National Opinion Research Center, and the Gallup Poll. For a detailed analysis of these data, see R. P. Quinn and L. J. Shepard, *The 1972–73 Quality of Employment Survey* (Institute for Social Research, University of Michigan, Ann Arbor, 1974).
7. A. Clayre, *Work and Play* (Harper & Row, New York, 1974).
8. R. Blauner, *Alienation and Freedom: The Factory Worker and His Industry* (Univ. of Chicago Press, Chicago, 1964).
9. T. L. Whisler, *The Impact of Computers on Organizations* (Praeger, New York, 1970). I. R. Hoos [in *Automation in the Office* (Public Affairs Press, Washington, D.C., 1961)] reaches more pessimistic conclusions than Whisler about the impact of the computer, but she mainly observed transient effects at the time the new technology was being introduced, and not the longer-term effects after the changes had been digested.
10. Two books that attack artificial intelligence on this ground are H. L. Dreyfus, *What Computers Can't Do* (Harper & Row, New York, 1972) and J. Weizenbaum, *Computer Power and Human Reason* (Freeman, San Francisco, 1976). The two books have little in common except a shared antipathy against the "machine" view of human thinking, and an eloquent contempt for those who hold that view. Weizenbaum's book is the technically more competent of the two, with respect to its understanding of the current state of the computer programming art.

# Trends in Computers and Computing: The Information Utility

Stuart E. Madnick

The complexity, interdependence, and rapidity of events in modern society have accelerated demands for more effective ways to store, process, and manage information. Advances in both computer hardware (electronics) and software (programming) have provided the technology that can make it possible to effectively address many of these demands. In this article I review the structure of classical computer-based information systems and then consider these advances and show how they fit into the evolution of the information utility.

### Information Systems

The need for more effective information management for both decision-making and operational efficiency is being felt for many reasons, such as:

1) Scarce resources. As we have become more aware of the scarcity of natural resources and the disruptions and perturbations in their supply (such as cost increases, boycotts, and strikes), it has become necessary to be much more responsive in decision-making in both the public and the private sectors.

2) Service productivity. There has been a growing demand for human services in most of the industrialized world. More than 66 percent of the American work force is employed in providing services; this includes teachers, lawyers, accountants, bank tellers, mailmen, and office workers. As noted by Hollomon (*1*), "The U.S. has evolved into a service economy and productivity is hard to manage in service-oriented industry. . . . Information technology is the most important element in a service economy." Information systems can improve the management of complex human services and provide greater operational efficiency, for example, through electronic funds transfer, electronic mail, automated office equipment, and electronic filing.

3) Complexity of society. As society becomes more complex, it is increasingly difficult to effectively assess the subtle interdependence of many decisions. As a result, "optimal" decision to resolve one problem may, in fact, precipitate worse problems in other areas. The frequency and significance of these counter-intuitive reactions are increasing (*2*) and this has prompted efforts to develop effective decision-support systems (*3*).

The element common to all of these concerns is information—to make decisions that can minimize the negative impacts of limited resources, to effectively manage the distribution of human services, and to cope with the complexity of society. Recent advances in computer and communications electronics, system design, and management science provide an increased potential for developing highly effective and efficient information systems.

The author is an associate professor of management science at the Alfred P. Sloan School of Management, a cofounder of the Center for Information Systems Research, and an adjunct member of the Laboratory for Computer Science (formerly Project MAC), all of the Massachusetts Institute of Technology, Cambridge 02139. This article is based on an internal report prepared in cooperation with Dean Witter & Co., Inc.

## Information Utility Evolution

As information system processing has become an increasing, if not the dominant, component of computer usage, it has become necessary to reexamine our traditional view of computer usage. In place of the "analytical engine" for numerical calculation, an information utility has evolved in which the computer is used primarily to assist in the generation, storage, retrieval, and distribution of information.

Let us consider a simple scenario illustrating the use of an information utility. On the request of the district sales manager of a large corporation, a staff member has prepared the district's sales forecast for the next 12 months. The scenario continues from that point.

1) A secretary, using a TV-like computer display terminal, types and edits the report. This process is often called computerized text processing or word processing.

2) The secretary then instructs the system to forward the report to the district sales manager. This process is called electronic mail. Note that the report remains in computerized form and is not necessarily typed on paper at any point.

3) The sales manager receives an indication on his display terminal that there is "mail" for him. When it is convenient, he examines the report on his terminal. Before sending the report out, he requests retrieval of the last few sales forecast reports to check for consistency. This process is usually called information retrieval or text retrieval. He instructs the system to file the new report for future reference by its date and under "sales forecast" and the author's name. He then forwards a copy to the vice president of sales at the corporate headquarters. In existing electronic mail systems, a five- to ten-page report can be delivered, even cross-country, in a few seconds.

4) After the vice president of sales has read the report, he forwards it to a member of his staff to prepare the corporate sales forecast. The staff member extracts the sales forecast information from all the district reports and also retrieves historical and economic data available from other sources. He then uses the computer to perform quantitative analysis and evaluate certain forecast models to arrive at a corporate sales forecast that can be used for production and financial planning.

The scenario, of course, could continue, but this brief excerpt should give the reader an idea of the operation of an information utility. There currently exist embryonic versions of such an information utility in several corporations and government agencies. Considerable research and the development of the information utility industry are still necessary before such a system becomes generally available.

Research on the technical and policy issues of telecommunications and electronic mail has been reported in several recent articles (*5–9*). This article will focus on the trends in computer technology and structure that can support the storage and retrieval aspects of the information utility.

## Information System Structure

Computer-based information systems are now being used for a wide array of applications, ranging from inventory control and production scheduling, to airline and rental car reservations, to energy policy analysis. The overall structure of a contemporary information system is depicted in Fig. 1. The basic computer system consists of the computer hardware, including the telecommunication lines and large-capacity storage devices, and the operating system software, which is the program used to control the basic hardware resources (*4*).

The information system software can be conceptually divided into four major components.

1) Telecommunications management system software handles communications with other computers and users at remote terminals.

2) Database management system (DBMS) software handles the general information storage and processing, is usually the most complex component, and in many ways is the heart of the system.

3) Application-dependent software performs specialized computations, such as evaluating a specific sales forecasting model. In many information systems there may be no need for specialized application-dependent software, or it may be handled by separate "personal" computers or "intelligent" terminals, as described later.

4) Information system manager software handles the coordination between the telecommunications, database, and application-dependent software as well as the interaction with the basic computer system.

The hardware and software requirements for information systems are outlined in the following sections with specific emphasis on the DBMS component.

## Key Hardware Components of a Computer System

The key hardware components of a typical conventional computer configuration that might be used to support an information system are shown in Fig. 2. The central processor fetches the machine instructions and data from the main storage (or main memory) and performs the designated operations, such as add two numbers. A sequence of instructions is called a program. The input/output (I/O) processors, operating under the direction of the central processor, handle the flow of information and commands between the main memory and the I/O devices such as card readers and printers. Communications controllers provide the interface to and controlling electronics for various forms of communication lines—for example, to remote terminals through the telephone network. Secondary storage controllers provide the interface to and controlling electronics for various secondary storage devices such as magnetic disks.

In response to the requirements for improved information management, advances in many areas of computer technology (*10*), especially microprocessors, microcomputers, large-scale integration (LSI) (*11*), memory devices (*12*), firmware (*13*), and communications (*6*), are making dramatically different computer hardware configurations desirable and feasible. Some advances that are particularly relevant to the evolution of the information utility are summarized below.

*Microprocessors and microcomputers.* A microprocessor or microprocessor unit contains the electronics of a central processor unit implemented in one integrated circuit (IC) package (or a small number of such packages). An IC package typically has an area of less than 2 square inches. The power supply and other auxiliary electronics, such as timing signals, are usually external to the microprocessor. The INTEL 8080A, currently the most popular 8-bit microprocessor, is a representative example. It primarily operates on a byte (8 bits) of information at a time (4-, 12-, and 16-bit microprocessors are also common). It has 78 instructions, can perform an add in 2 microseconds, and sells for less than $20 in quantity.

A microcomputer contains the electronics of the central processor unit plus some or all of the main storage in one or a few IC packages. The newly announced INTEL 8748, for example, is conceptually similar to the 8080A but contains the microprocessor unit, 1024 (1K) bytes of reprogrammable read-only

memory, 64 bytes of read/write memory, 27 I/O lines, a timer/counter, and most other system necessities on a single IC. The 8748 is expected to sell for less than $100 in quantity.

Using microprocessor and microcomputer technologies, inexpensive yet powerful computer systems can be developed. Digital Equipment Corporation's LSI-11, for example, is basically a microprocessor-based version of its PDP-11 16-bit minicomputer. The LSI-11, with 2K bytes of memory, more than 400 instructions, and an add time of 3.5 $\mu$sec, is available mounted on an 8½ by 10 inch circuit board with various auxiliary electronics for less than $1000 in quantity.

In contrast, the ENIAC computer, completed in 1946, had 18,000 vacuum tubes, required 15,000 square feet of floor space, weighed 30 tons, consumed 150 kilowatts of electricity, and cost $400,000, yet could only perform about 5000 additions per second and needed to be programmed by physical wire connections (*14*).

The microprocessor and microcomputer industry has been growing very rapidly. As noted in the 1976 "Microcomputer systems directory" (*15*), "Roughly 20% of the companies listed *didn't even exist* last year, while about 25% are garage-sized microcorporations."

The evolution of microprocessors and microcomputers has made it economically possible to distribute more "intelligence" throughout the computer configuration, especially in the devices, controllers, and I/O processors. Furthermore, in some cases the traditional central processor itself is being replaced by "a federation of microprocessors" (*16*), as in IBM's System/370 models 115 and 125.

As an interesting side effect of the rapidly decreasing costs of LSI processors and main memories, the costs of the other ingredients of a computer (power supplies, cooling units, cabling, cabinets, and so forth) are becoming the major ones, and are estimated to increase from about 33 percent at present to 77 percent by 1980 for conventional computers (*17*). This phenomenon makes the widespread use of processors and larger memories even more attractive economically.

*Memory technologies.* Traditionally, computer direct-access storage has been dominated by two fairly distinct technologies: (i) ferrite core and, later, metal oxide semiconductor LSI memories with microsecond access times and relatively high costs, and (ii) rotating magnetic media (magnetic drums and disks) with access times in the range 10 to 100 milliseconds and relatively low costs. This has led to the separation between main storage and secondary storage noted earlier.

The evolution and increasing deployment of various new memory technologies have produced a more continuous cost-performance range of storage devices, as depicted in Table 1 (*10, 18–20*). Note that these technologies, which are arbitrarily grouped into six categories, result in storage devices that span

Table 1. Range of storage devices.

| Storage level | Random access time (seconds) | Sequential transfer rate ($\times 10^6$ byte/sec) | Unit capacity (bytes) | System price (cents per byte) | Technology |
|---|---|---|---|---|---|
| Cache | $100 \times 10^{-9}$ | 100 | $32 \times 10^3$ | 50 | Bipolar LSI random-access memory |
| Main | $1 \times 10^{-6}$ | 16 | $512 \times 10^3$ | 10 | Metal oxide semiconductor LSI random-access memory, ferrite core |
| Block | $100 \times 10^{-6}$ | 8 | $2 \times 10^6$ | 2 | LSI shift registers, bulk ferrite core, charge-transfer devices magnetic bubbles |
| Backing | $2 \times 10^{-3}$ | 2 | $10 \times 10^6$ | 0.5 | Fixed-head magnetic disks and drums, charge-transfer devices, magnetic bubbles, electron beam |
| Secondary | $25 \times 10^{-3}$ | 1 | $100 \times 10^6$ | 0.02 | Moving-head disks |
| Mass | 1 | 1 | $100 \times 10^9$ | 0.0005 | Automated tape handlers, optical (laser) beam |

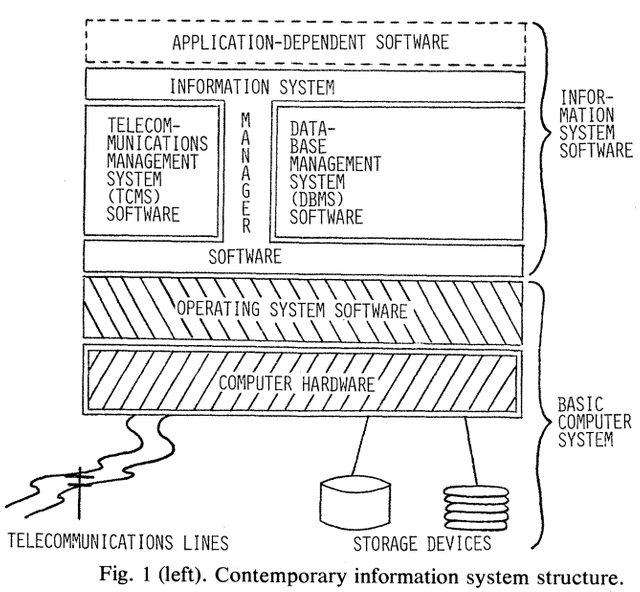



Fig. 1 (left). Contemporary information system structure.

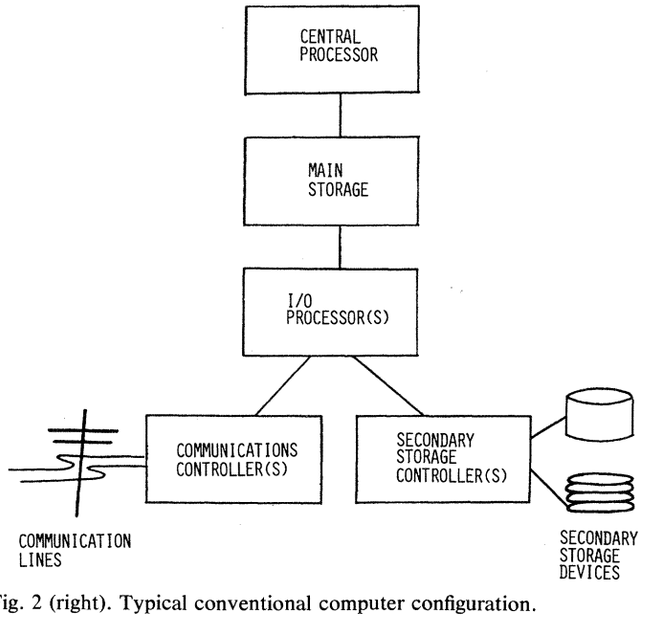



Fig. 2 (right). Typical conventional computer configuration.

Table 2. Programming languages.

*A. Sample database:* EMPLOYEES

| NAME | OFFICE# | EXT# | DEPT# | JOB-TITLE |
|---|---|---|---|---|
| ADAMS, JOHN | E53-330 | 3-7156 | 15 | TECHNICIAN |
| BROWN, TOM | 39-604 | 3-6711 | 15 | MANAGER |
| DOUGLAS, JOE | E52-154 | 3-2116 | 14 | ENGINEER |
| . . . | . . . | . . . | . . . | . . . |

*B. Database query language example*

```
PRINT NAME AND EXT# FOR ALL EMPLOYEES
WHERE DEPT#=14 AND JOB-TITLE=ENGINEER;
```

*C. Conventional high-level programming language equivalent*

```
        OPEN FILE (EMPLOYEES) INPUT;
        ON ENDFILE (EMPLOYEES) GOTO DONE;
NEXT:   READ FILE (EMPLOYEES) INTO (EMPLOYEE);
        IF EMPLOYEE.DEPT#=14 & EMPLOYEE.JOB-TITLE='ENGINEER'
        THEN PRINT (EMPLOYEE.NAME, EMPLOYEE.EXT#);
        GOTO NEXT;
DONE:   CLOSE FILE (EMPLOYEES);
```

*D. Conventional low-level (assembly) language equivalent*

```
        OPEN    FILE=EMPLOYEES, MODE=INPUT, ENDFILE=DONE
NEXT:   READ    FILE=EMPLOYEES, TYPE=SEQUENTIAL, AREA=EMPLOYEE
        CLC     EMPDEPT,=F'14'
        BNE     NEXT
        CLC     EMPTITL,=C'ENGINEER'
        BNE     NEXT
        MVC     OUTNAME, EMPNAME
        MVC     OUTEXT#, EMPEXT#
        WRITE   FILE=SYSPRINT, TYPE=SEQUENTIAL, AREA=OUTPUT
        B       NEXT
DONE:   CLOSE   FILE=EMPLOYEES
```

more than six orders of magnitude in both random access time (from less than 100 nanoseconds to more than 1 second) and system price per byte (from more than 50 cents to less than 0.0005 cent). This phenomenon blurs the distinction between main and secondary storage and is forcing a serious reappraisal of traditional computer architecture. There is a significant trend toward an automatically managed multilevel storage hierarchy (*18*).

Furthermore, the information in Table 1 shows that a single unit of mass storage can store and provide rapid access to the equivalent of 25 million pages of single-spaced text (assuming about 4000 characters per page) for a cost of under 2 cents per page. Coupled with anticipated further cost reductions, this provides a tremendous potential for displacing paper for many uses, leading to electronic filing and electronic mail.

## Information System Software

Information systems, especially the DBMS component, provide for convenient storage, organization, restructuring, and access to information stored in the computer system.

Table 2A illustrates a simple database, named EMPLOYEES, containing employee information, in particular each employee's name, office number, telephone extension number, department number, and job title. By using a database query language, a request to see "the names and extension numbers of all engineers in department 14" can be easily expressed in a very high-level English-like manner, as depicted in Table 2B. In contrast, such a request in conventional high-level programming languages such as Cobol, Fortran, or PL/I (Table 2C) or conventional low-level assembly languages (Table 2D) would be much less user-oriented and lengthier.

This example, although it may be informative, understates the capabilities and operational requirements of most modern database systems. Instead of handling a single type of information, as in Table 2, it is often necessary to handle hundreds of different types of information, each consisting of the equivalent of hundreds of thousands of "records," as well as the complex relationships between them. For example, consider a system containing manufacturing information on parts inventory, production schedules, customers information, sales orders, and so on. A request to such a system might be, "List all customers in the northeast sales region whose current outstanding orders cannot be satisfied by our inventory on-hand or planned production within 30 days."

The services provided by most database management systems can be divided into two categories: (i) physical storage management, which is concerned with the characteristics of the storage devices, and (ii) logical storage management, which is concerned with the structure of the information, especially as seen and manipulated by the user.

*Physical storage management.* To economically hold large databases, it is necessary to use storage devices with a low cost per byte, such as moving head disks or tape cartridges (see Table 1). But since these devices have fairly slow access, higher-speed devices, such as main memory, are typically used to temporarily hold (buffer) active portions of the database. In this way, many requests for information can be satisfied at high speed by using the main memory buffers without needing to access the slower secondary storage. This system is called locality of reference (*21*) and is widely used in database systems. (It is similar to having a bookcase of frequently used books so that it is only occasionally necessary to go to the larger, but less convenient, public library.) Since the locus of activity in a database will usually shift in time, the physical storage management component of a DBMS must keep track of the current location of all information and move information among the various storage devices as necessary to maintain high performance.

If a variety of storage devices such as those shown in Table 1 are used, physical storage management may attain very high levels of cost-performance effectiveness but, in the process, may incur considerable system overhead. Thus, many hardware and software strategies are being investigated to increase the sophistication of physical storage management while reducing its operational overhead. Several of the major approaches are discussed later in this article.

*Logical storage management.* Even if there were a single storage device that was extremely fast, had enormous capacity, and was very inexpensive (which is very unlikely in the near future), DBMS logical storage management functions would still be needed. These functions are multitudinous but most can be placed into two categories: (i) performance-related and (ii) mapping-related.

Many DBMS functions are related to searching, as exemplified by Table 2B. A linear top-to-bottom search, as in Table 2, C and D, is conceptually very simple but can be very inefficient for large databases. For example, even if each step of a linear search could be performed in 10 $\mu$sec, it could take more than 15 minutes to locate a specific record in a database of 100 million records. Although that is still quite fast by human standards, many database systems with currently avail-

able storage devices (such as moving-head disks with 25-msec access) can accomplish such a feat in less than 1 second. This is done through various more sophisticated searching procedures (for example, binary search or "hash coding"), usually in conjunction with auxiliary data structures (for example, alphabetically ordered indexes or cross-reference lists). As the number of uses for and users of such databases increases, it becomes necessary to handle hundreds or thousands of such queries per minute. Thus, there are many software and hardware advances directed at improving the performance of DBMS logical storage management.

Although performance is important, ease of use is critical to the future of database systems. For example, the DBMS must be able to accept a high-level nonprocedural query, such as that in Table 2B, and map it into the basic functions provided by the underlying hardware (typically at the level illustrated in Table 2D).

For the DBMS to perform these mappings automatically and correctly, considerable information *about* the information stored in the database must be maintained in various forms of data dictionaries. This may range from format information, such as how many digits are in an extension number, to data validation and editing information, such as what are allowable department numbers, to privacy and security constraints, such as prohibiting exmployees from changing their own salaries.

Logical storage management and especially the mapping functions have played a key role in making information systems easier and more effective to use. These features do, however, consume considerable resources. In many cases, the mapping activities may take more central processor time than the actual data retrieval or manipulation requested by the user. Thus, this is an area of considerable focus for software and hardware advances.

## Specialized Information System Machines

The basic design of today's computers has remained largely unchanged for more than two decades. The components have become faster, smaller, and cheaper, but the overall structure is still primarily focused on mathematical calculation rather than the storing, retrieving, and data organizing functions that computers are actually being used for today. As Mueller (*22*) noted, "The computer industry has gone through three generations of development to perfect machines optimized for 10 percent of the workload."

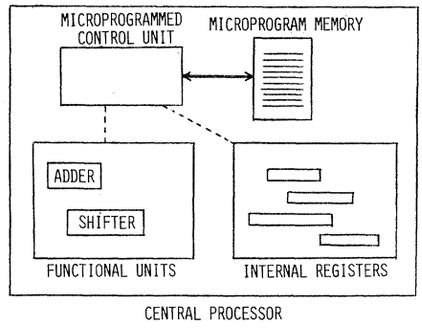



Fig. 3. Microprogrammed central processor.

The time is now ripe for a serious reexamination of the conventional computer, as exemplified by Fig. 2, especially in light of the hardware and software advances attained or forecasted. Eckert (*14*), one of the developers of the ENIAC, said, "I have often wondered why somebody didn't invent the ENIAC 10 years earlier. The tubes were almost as good then. . . . Therefore I have to really believe, at least for most of my experiences, that necessity is usually the mother of invention. . . ." It is clear that we are confronted with necessity once again; what is needed now is the imagination and leadership to apply the basic technologies to the critical problem of information management.

In this section various efforts along these lines are presented. The approaches can be largely classified into four categories: (i) firmware enhancements, (ii) "intelligent" controllers, (iii) minicomputer "back-end processors," and (iv) highly modular database machines.

*Firmware enhancements.* Most modern computers (as in Fig. 2) use the approach of microprogramming in the design of the central processor unit (*23*). That is, the central processor's internal registers and functional units used to decode and execute the computer's instructions are controlled by a much more primitive microprogrammed control unit carrying out sequences defined in the high-speed microprogram memory (see Fig. 3). For example, the microprogram determines what machine instruction will be interpreted to mean multiply and how the multiply operation is actually accomplished, typically by means of a number of shifts and additions. Thus, it is the contents of the microprogram memory that determines the computer's instructions as seen by a "machine language" programmer. This approach is often called microprogramming, microcoding, or firmware. By using various different microprograms, the same hardware may take on the appearance of various different computers; this is usually called emulation.

Conventional computer instructions are usually not well suited to the requirements of operating systems and database systems. Using firmware, it is possible to augment or enhance the instructions. Such an approach, called virtual machine assist, has been exploited extensively to support operating system functions in the recent IBM System/370 model 138 and 148. Similar operating system support techniques have been successfully used by many other computer manufacturers.

Firmware can be used to enhance otherwise conventional computers to make them more suitable for supporting database processing. Instead of using inefficient subroutines, each database operation may be executed as a single microprogrammed instruction. This approach has been adopted in several systems. One of the earliest efforts was part of the Lincoln Information Storage and Retrieval (LISTAR) system developed at the Massachusetts Institute of Technology (MIT) (*24*), in which several frequently used operations, such as a generalized search list operation, were incorporated into the microcode of an IBM System/360 model 67 computer. The recently announced Honeywell H60/64 has special instructions to perform data format conversion and "hashing" corresponding to frequently used subroutines of Honeywell's IDS database system (*25*). The performance advantages of this approach are highly dependent on the frequency of use of the new instructions and the extent to which they fit into the design of the overall database system software.

A more extreme approach is to define a completely new set of computer instructions through firmware. In the past, emulators have been developed that directly execute high-level languages, such as APL (*26*). The Microdata Reality System uses this approach to implement a complete database system in the microcode of an otherwise conventional minicomputer.

Since the basic hardware remains unchanged, the performance improvement achieved with firmware is largely in the higher speed of the microprogram memory (which is typically two to ten times that of main memory) and the ability of microprograms to perform certain basic operations, such as bit manipulations, very efficiently. On the other hand, since microinstructions are usually quite primi-

tive, it may take many of them to perform a specific function. With complete firmware emulation, the overall improvement in central processor performance is typically in the range of 50 to 500 percent (*26*)

*Intelligent controllers*. Many physical or logical storage management functions can be accomplished by intelligent secondary storage controllers (controllers containing microprocessors or microcomputers). For example, based on the concept of locality of reference, there have been various efforts to automate physical storage movement between devices, resulting in cache memory and virtual memory systems (*18, 27, 28*). In the IBM 3850 Mass Storage System (*29*), for example, an intelligent controller is used to automatically handle the transfer of data between high-capacity, slow tape cartridges and medium-capacity, faster moving-head disks (Fig. 4a). From the point of view of the user (that is, the central processor), the 3850 appears as a large number of disk units (Fig. 4b).

The 3850's real disks are used to hold active portions of the database, typically in units of cylinders (approximately 250,000 bytes). The controller maintains a record of the cylinders that are currently stored on the disks. When a request is received to read data from, say, virtual disk 2416 cylinder 36, the controller checks its storage location table to see if that cylinder is currently on one of the real disks; if it is, the requested data are immediately transferred to main storage. If not, the corresponding tape cartridge (for example, cartridge 4832) is fetched and scanned to locate the correct cylinder, the entire cylinder of data is then transferred to an available cylinder on one of the real disks, and the storage location tables are modified correspondingly. The requested data are then transferred from the disk to main storage as described above.

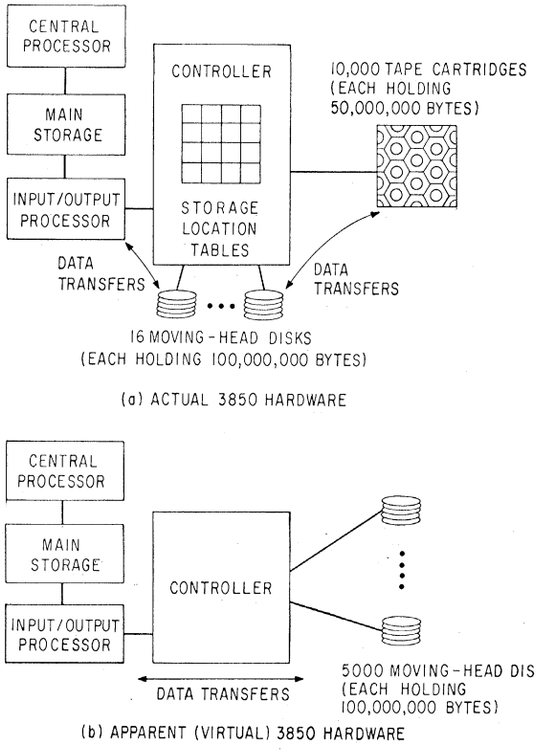
When the requested data are already on the real disks, the request can be satisfied in about 25 msec; otherwise it may take 5 to 10 seconds. Except for this possible delay, the 3850 functionally performs exactly like 5000 disks but at only 2 to 3 percent of their cost. For many applications, where only small localized portions of a very large database are used at any time, the overall performance may be almost the same.

Many storage controllers have been designed to perform certain logical storage management functions (*30*). For example, simple single-key linear search is a common feature. Linear search, even if done by the controllers rather than the central processor, is generally quite inefficient for large databases. In order to gain much higher performance, experimental controllers that perform parallel "associative" search strategies have been developed (*31–33*).

Most parallel associative search controller strategies are based on a "head per track" storage device technology (for example, magnetic drum, LSI shift registers, and magnetic bubbles) and a multitude of comparators, as depicted in Fig. 5. As each data record "rotates," either mechanically or electronically, past a read/write head, it is compared with a match record register, called the mask. If there is one comparator per head per track, as in Fig. 5, 1000 comparisons are done simultaneously; in this example a database of 160,000 100-byte records could be searched in about 10 msec, the rotation time for the storage device. In terms of conventional sequential search technologies, this is equivalent to reading and comparing one record every 60 nsec. In addition, the comparisons may be fairly complex, involving multiple data fields of the record (for example, searching for DEPT#=14 AND JOB-TITLE=ENGINEER in Table 2).

At present, the cost of the comparators and read/write heads limits this approach to medium-size databases (10 to 100 million bytes). Furthermore, it is only well suited to storage technologies that lend themselves to low cost read/write mechanisms, and for optimal performance and operation it tends to require a fairly simple and uniform database structure.

The decline in the costs of comparator electronics, due to advances in LSI technology, makes parallel search strategies quite promising for the future. Variations on this approach are currently in use for certain real-time applications requiring



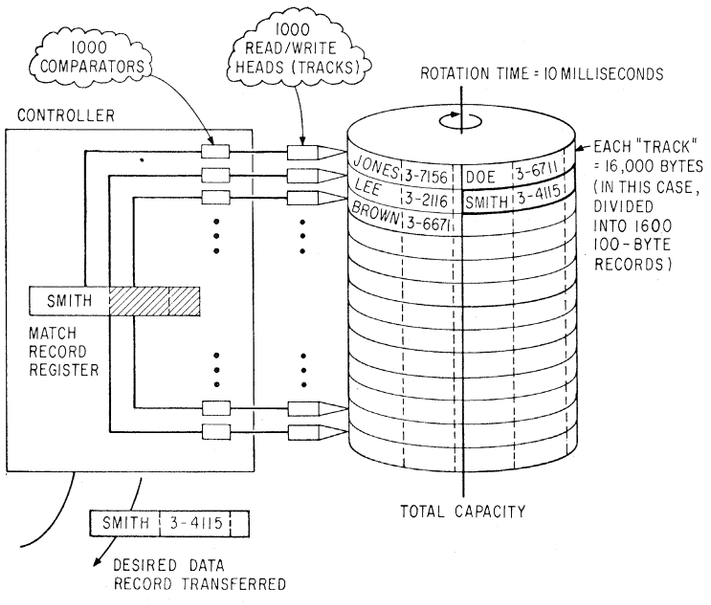




Fig. 4 (left). Use of intelligent controllers for physical storage management. Fig. 5 (right). Sample parallel associative search controller. Total capacity = 1000 tracks × 16,000 byte/track = 16,000,000 bytes (160,000 100-byte records).

high performance, such as missile tracking.

*Back-end processors*. As an extension of the intelligent controller approach, we can remove all of the functions of a traditional database system from the central processor and incorporate them in a back-end processor (*34*), as depicted in Fig. 6. (Front-end processor is an earlier term used to describe sophisticated communications processors.)

The back-end processor is usually a minicomputer specifically programmed to perform all (or most) of the functions of the database management system. Although such a system could be operated independent of any other computer (stand alone), we will focus our attention on the cases where the back-end processor is connected to other processors or computers, either directly to physically close systems (tightly coupled) or through a communications system to physically remote systems (loosely coupled or distributed).

There are many advantages to this approach, including the following.

1) Low processor cost. Because of the processing characteristics of database management software, a low-cost high-performance minicomputer may perform as well as (or better than) a high-cost traditional processor optimized for mathematical calculations. (For example, the ability to perform a 64-bit floating point multiply in 1 $\mu$sec is not generally needed in database processing.) By removing the database software load, the central processor can be more fruitfully used to service the remaining work load. The amount of load removable may be 40 to 50 percent or more (*35*).

2) Sharing and distributed database processing. Back-end processors can serve as shared repositories of information accessible by all of the connected computers, while at the same time enforcing defined information security constraints (*36*). The Datacomputer (*37*), operating as a loosely coupled back-end processor through the Arpanet, an international communications network, can serve as the information repository for dozens of computers. In such an environment, multiple back-end processors, each serving as a database node, may be connected to provide distributed database processing capabilities.

3) Low storage cost. By pooling the storage requirements of many computers, more economical high-volume storage devices may be used. Many of the minicomputers and specialized processors on the Arpanet use the Datacomputer because of its large capacity and low storage cost per byte.

4) Compatibility and uniformity. A large corporation or government agency may have dozens of different computers (often from different manufacturers). Interchange of information between such computers has usually been awkward because of incompatibilities between the computers and lack of uniformity in the database. By use of the back-end processor concept, the information may be conveniently shared or interchanged among these disparate computers.

Back-end processors have evolved rapidly in recent years. Some of the earliest experimental efforts include the loosely coupled Datacomputer (*37*) developed by the Computer Corporation of America using the DEC System-10 computer, and the tightly coupled XDMS (*34*) developed by Bell Laboratories by modifying the firmware of a Digital Scientific META-4 minicomputer.

The commercialization of back-end database processors is proceeding down various paths, offering different advantages to the user. As an example, Cullinane Corp., developers of the IDMS database system software for large-scale computers, is under contract with various government agencies to deliver a back-end processor, based on the PDP-11/70 minicomputer, to be attached to IBM System/360 and System/370 computers (*38*). This facility may be very attractive to users of the older System/360. Relieving the 360, often a purchased rather than rented computer, of the database processing makes additional capacity available for the increasing application-dependent production work load. This can extend the usable lifetime of the installed computer and avoid a possibly costly conversion of the production programs if it were necessary to upgrade to new computers.

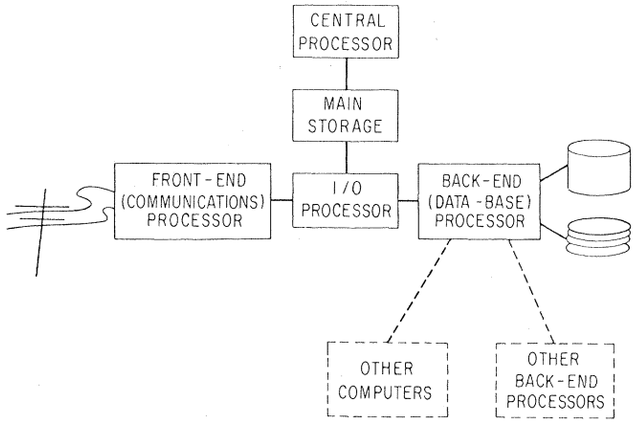



Fig. 6 (left). Back-end processors. schematic.

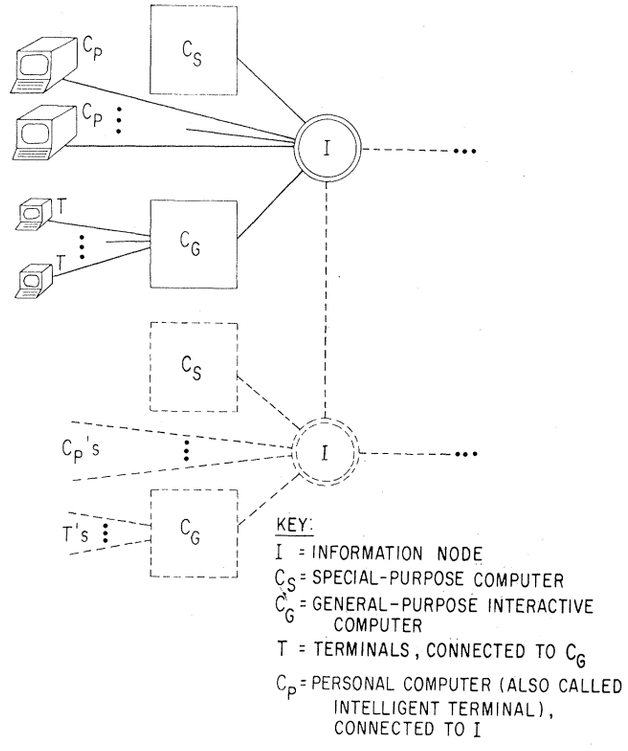



Fig. 7 (right). Information utility

It is expected that many more commercial back-end processor systems will become available in the next few years.

*Database machines.* The back-end processors described above were based on carefully selected and, occasionally, slightly modified conventional computers. This approach can provide significant cost-performance advantages, possibly up to tenfold, over the traditional use of the central processor for both applications programs and database processing. But there still remain major limitations on performance and reliability capabilities.

Typical contemporary information systems handle one to five requests (or transactions) per second. A system is usually viewed as having a very high volume if it needs to (and is able to) satisfy 50 to 100 requests per second. If the demand originates only from a finite number of remote terminal users (say 2000), human typing and thinking time ($\sim$ 20 seconds per request) would probably keep the load on the system within those limits. But consider the performance that would be needed for a back-end processor to serve as an information repository to a network of 100 or 1000 computers, both large-scale and personal, each capable of generating hundreds or thousands of requests per second. Such a situation—a likely one to arise in the evolution of the distributed processing and information utility concepts—could result in request rates of more than 1 million per second. This is well beyond the capabilities of conventional computer architecture.

Despite many advances in component reliability and the use of error-checking procedures, most large-scale computer systems are very complex and have many weak spots where a failure can occur and cause the entire system to stop. A mean time between failures of 48 hours is not uncommon for many large-scale commercial computers. If the back-end processor is to service a network of users, both humans and computers, such an interruption of service could be extremely disruptive. This type of concern has led to the development of various specialized "no stop" highly redundant multiple processor systems, such as the Tandem minicomputer system and the Pluribus communications system (*39*). Conventional computer architectures are not adequate to provide the necessary degree of reliability.

Probably the most promising approach to attain both high performance and reliability is to decompose the database system into its underlying functional modules (both physical and logical storage management functions) and use a multiplicity of microprocessors to allow significant parallel processing of each function.

The DBC database computer design (*36*) uses specialized components, many based on associative search mechanisms, to handle the storage of data dictionary information, processing of directory information, storage of the database, and security enforcement.

The Infoplex design (*18, 40*), in which physical and logical storage management functions are explicitly separated, defines more than 12 separable functional levels. By using up to 4000 microprocessors, with 100 or more in parallel at each functional level, extremely high reliability and performance can be attained (all processors for a functional level are anonymous and operate under local control; if one fails, performance is reduced slightly, but operation continues uninterrupted).

Much additional research and experimentation in the design and development of database computers will be needed to meet the performance and reliability requirements of the information systems of the 1980's.

## Information Utility Structure

As the evolution of the information utility continues, there will be an increased separation between computer systems specialized for reliable, economical storage and processing of large amounts of information, called information nodes, and systems specialized to support traditional numerical computation, as well as personal computer usage. Figure 7 is a view of such an information utility. Information nodes may be interconnected, resulting in a geographically distributed database. An information node may be connected to various types of computers, as follows.

1) Special-purpose computers. These may be specialized in their construction for high-performance numerical computation (for example, a signal processor) or in their function (such as process control).

2) General-purpose interactive computers. These correspond to most contemporary computer systems that are used to support specific application algorithms (such as sales forecasting models) as well as program development. Users can communicate with the computer through remote terminals.

3) Personal computers. Advances in minicomputers and sophisticated terminals have spawned a movement toward distributed processing, in which calculations, word processing, and other such tasks are performed by personal computers (or intelligent terminals) rather than a more complex general-purpose interactive computer.

In all of these cases, the connected computers can draw on the information node for needed data or deposit new data, either gathered or generated. The Generalized Management Information System at MIT, for example, uses such an arrangement of virtual computers in support of the New England Management Information System (*41*) project. The purpose of the project was to develop an information system for the New England Regional Commission to assist policy-makers in managing the problems brought on by the energy crisis. Better information was needed to analyze methods to conserve fuel and to assess the impacts of tariffs and prices on different industrial sectors and states within the region. Support was also required for many other studies, such as the analysis of the merits of refineries and the impact of offshore drilling on the fishing industry. The computer-based NEEMIS project was developed to allow researchers to respond to such demands for information that had not been previously anticipated. Positive results have been reported from this project, and much can be learned from its example. Other such systems are in the development stage.

## Conclusions

Various studies indicate that the installed value of computers, worldwide, will double in the next 5 years; the primary emphasis will be on a general transition from traditional batch processing to transaction processing, providing rapid response from central databases through a communications network (*42*). The evolution of the information utility concept and the accompanying technical advances are the keys to meeting these demands.

As observed thus far, information utilities will most likely be implemented first within large governmental and industrial organizations to serve their own needs. Depending on favorable governmental regulations on telecommunications and the successful implementation of the information node technologies outlined in this article, it is possible that an information utility industry, offering services to general subscribers, will be established by the mid-1980's.

Although the development of computer hardware and system software is im-

portant, it must be realized that the full potential of the information utility depends on dramatic changes in information system usage and structure (for example, electronic mail, electronic filing, automated office operation, and electronic funds transfer). In many cases, these changes are more under the control of information system designers and users than of computer manufacturers. An informed user community is an important requirement for future progress.


### Summary

Demands for more effective information management, coupled with advances in computer hardware and software technology, have resulted in the emergence of the information utility concept, whereby computers specialized for information storage and processing serve as information nodes. The information nodes, which may be interconnected, can provide information management services to both conventional and personal computers. In this article the key hardware and software components of classical information systems are described to provide background on the requirements for an information utility. Four approaches to the development of specialized information nodes, drawing on various advances in technology, are presented: (i) firmware enhancement, (ii) intelligent controllers, (iii) minicomputer back-end processors, and (iv) highly modular database machines. The benefits of these advances will be systems that are more efficient, reliable, and easy to use.


#### References


1. H. Hollomon, *Technol. Rev.* **77**, 57 (January 1975).
2. J. W. Forrester, "Dynamics of socio-economic systems," Report D-2230-1, MIT Systems Dynamics Group, Cambridge, Mass., August 1975.
3. J. J. Donovan and S. E. Madnick, *Database* **8** (No. 9), 79 (1977).
4. S. E. Madnick and J. J. Donovan, *Operating Systems* (McGraw-Hill, New York, 1974).
5. B. Edelson, *Science* **195**, 1125 (1977).
6. D. Farber and P. Barran, *ibid.*, p. 1166.
7. M. Irwin and S. Johnson, *ibid.*, p. 1170.
8. S. E. Miller, *ibid.*, p. 1211.
9. R. J. Potter, *ibid.*, p. 1160.
10. W. Myers, *Computer* **9** (No. 11), 48 (1976).
11. R. Noyce, *Science* **195**, 1102 (1977).
12. J. Rajchman, *ibid.*, p. 1223.
13. H. D. Mills, *ibid.*, p. 1199.
14. J. P. Eckert, *Computer* **9** (No. 12), 58 (1976).
15. J. Conway and P. Snigier, *EDN* **21** (No. 21), 95 (1976).
16. H. D. Toong, *AFIPS Conf. Proc.* **44**, 567 (1975).
17. S. E. Madnick, *Technol. Rev.* **75**, 8 (July/August 1973).
18. ———, *IEEE Intercon Conf. Rec.* (1975), pp. 20/1-1 to 20/1-7.
19. R. R. Martin and H. D. Frankel, *Computer* **8** (No. 2), (1975).
20. J. H. Wensley, *ibid.*, p. 30.
21. P. J. Denning, *ACM Comput. Surv.* **2** (No. 3), 153 (1970).
22. G. Mueller, *Computer* **9** (No. 12), 100 (1976).
23. S. H. Fuller, V. R. Lesser, C. G. Bell, C. H. Kaman, *IEEE Trans. Comput.* **C-25** (No. 10), 1000 (1976).
24. A. Armenti, S. Galley, R. Goldberg, J. Nolan, A. Sholl, *AFIPS Conf. Proc.* **36**, 313 (1970).
25. C. Bachman, *ibid.* **44**, 569 (1975).
26. A. Hassitt and L. E. Lyon, *IBM Syst. J.* **15** (No. 4), 358 (1976).
27. R. L. Mattson, J. Gecsei, D. R. Slutz, I. L. Traiger, *ibid.* **9** (No. 2), 78 (1970).
28. R. M. Meade, *AFIPS Conf. Proc.* **37**, 33 (1970).
29. C. Johnson, *ibid.* **44**, 509 (1975).
30. G. R. Ahearn, Y. Dishon, R. N. Snively, *IBM J. Res. Dev.* **16** (No. 1), 11 (1972).
31. S. Y. Su and G. J. Lipovski, in *Proceedings of the International Conference on Very Large Data Bases* (Association for Computing Machinery, New York, 1975), pp. 456–472.
32. E. A. Ozkarahan, S. A. Schuster, K. C. Smith, *AFIPS Conf. Proc.* **44**, 379 (1975).
33. S. C. Lin, D. C. P. Smith, J. M. Smith, *ACM Trans. Database Syst.* **1**, 53 (March 1976).
34. R. H. Canaday, R. D. Harrison, E. L. Ivie, J. L. Ryder, L. A. Wehr, *Commun. ACM* **17** (No. 10), 575 (1974).
35. H. C. Heacox, E. S. Cosloy, J. B. Cohen, in *Proceedings of the International Conference on Very Large Data Bases* (Association for Computer Machinery, New York, 1975), pp. 511–513.
36. R. I. Baum and D. K. Hsiao, *IEEE Trans. Comput.* **C-25** (No. 12), 1254 (1976).
37. T. Marill and D. Stern, *AFIPS Conf. Proc.* **44**, 389 (1975).
38. J. Verity, *Electron. News* (3 January 1977), p. 28.
39. S. M. Ornstein, W. R. Crowgher, M. F. Kraley, R. D. Bressler, A. Michel, F. E. Heart, *AFIPS Conf. Proc.* **44**, 551 (1975).
40. S. E. Madnick, *ibid.*, p. 581.
41. J. J. Donovan, *ACM Trans. Database Syst.* **4** (No. 1), 344 (1976).
42. F. G. Withington, *Datamation* **21** (No. 1), 54 (1975).


# Software Engineering

## A mathematical basis is needed for the practical control of computers in complex applications.


Harlan D. Mills


Computer software began as an afterthought to computer hardware, and, as long as the hardware was small and simple, software could be handled informally by scientifically trained people as a by-product of the use intended for the hardware. As hardware grew in size and complexity, richer software possibilities emerged and software specialists (programmers) appeared, to produce assemblers, compilers, operating systems, and data management systems. Although there was an early recognition of mathematical ideas in computing, for example, in mathematical logic, automata theory, and linguistics, the bulk of these software specialists were pragmatic products of computing practice rather than mathematicians. Thus, although it may seem surprising, the rediscovery of software as a form of mathematics in a deep and literal sense is just now beginning to penetrate university research and teaching, as well as industry and government practices. The forcing factor in this rediscovery has been the growth of software complexity and the inability of informal software practices and management to cope with it.

Of course, software makes totally new demands in the sheer volume of logical precision required in its application. A single project may occupy hundreds, even thousands, of people over several years, so that there are unique requirements for recording, communication, and management of the work. These unique requirements lead to almost all of the jargon in software, and, in fact, this jargon tends to obscure the mathematical character of software, as people get caught up in implementation and management details. But the failure to understand this mathematical character led to an overly complex, ad hoc view of software based on historical and accidental ideas, which were often reinvented in ignorance and haste.

The work of Dijkstra and Hoare has been a major force in this rediscovery of software as mathematics. In (*1*, p. 4.2), Dijkstra has presented an argument which sums up the case I want to make here:

> As soon as programming emerges as a battle against unmastered complexity, it is quite natural that one turns to that mental discipline whose main purpose has been since


The author is an IBM Fellow at International Business Machines Corporation, Gaithersburg, Maryland 20760. He also teaches computer science part time at the University of Maryland, College Park.