Management Services: A Magazine of Planning, Systems, and Controls

Volume 7 | Number 4 Article 8

7-1970

# Software and the General Manager

William E. Lindsay

*The recent rash of 'unbundling' decisions by EDP manufacturers and constantly rising costs have propelled 'software' into the consciousness of both managers and consultants. Here are some aspects to be considered—*

# SOFTWARE AND THE GENERAL MANAGER

*by William E. Lindsay*
*Supply Corps, U.S. Navy*

TODAY'S general manager is bombarded with literature on how to and how not to select an EDP system—buy an EDP system—install an EDP system—manage an EDP system—and modify an EDP system. (The term general manager as used here does not mean the direct manager of an EDP installation but rather the supervisor or supervisors above him with responsibility areas including but not limited to EDP. Examples would be the executive vice president of a company where the EDP manager is in a staff position reporting directly to top management; a comptroller who not only manages the EDP installation through an EDP manager but also is responsible for the organization's budgeting and financial resources; or the production manager who manages the EDP installation as well as the major output process of an organization.[1])

One example of the information avalanche concerning EDP problems, an article by J. Richard Sherman in *Data Processing Magazine,* begins by asking the following questions:

Would computers be a profitable investment for our company? What applications would bring the greatest profit? What is a management information system, and should we be moving toward implementing such a system? What should we do, buy or lease? What will our future

[1] It is granted that most management specialists and consultants would not cut the actual EDP manager off from top management contact by having him report to a line or staff executive; however, it is sometimes done. See Marvin M. Wofsey, *Management of Automatic Data Processing Systems,* Thompson Book Company, Washington, D.C., 1968, pp. 21-38, and Dick H. Brandon, *Management Standards for Data Processing,* D. Van Nostrand Company, Inc., Princeton, New Jersey, 1963, pp. 30-31, for recommended organizational patterns.

The views expressed in this article are those of the author, not those of the Department of Defense.

**Software is analogous to the support forces needed to keep a fighting unit in the field**

information needs be, and how do we propose to meet them?[2]

This article, however, focuses on some aspects of what the general manager should know about a part of the EDP system not mentioned by Mr. Sherman, the software, the non-hardware. Software is analogous to the support forces that must exist to keep an armored unit in the field or a ship at sea. In its relationship to hardware it is similar to an iceberg. In the military example, that part of the iceberg above the ocean is the combat force, and the submerged portion depicts the support force. In EDP the above-water area is the hardware, the computer, and the submerged part would be the computer software.

### *Hardware knowledge widespread*

In the past five or ten years most general managers have acquired some familiarity with computer hardware. Like anything with physical shape and substance, hardware can be visualized. In most minds this creates an image, which, by the way, may or may not correspond to reality. Many general managers also have a knowledge of leasing arrangements, equipment costs, and operator requirements.

Nevertheless, when software is mentioned, some general managers realize that they have serious gaps in their knowledge. For, like the word itself, software is conceptually soft and difficult to grasp. Yet the technical capabilities of the general manager's hardware and the application of computers to his problems can be greatly hampered if he lacks understanding of software. Many general managers are harassed with the gnawing, and in many cases soon confirmed, realization that equipment cost, space cost, and management cost are only a fraction of the total cost of their EDP installation. They soon become aware that the omnivorous beast of software has an insatiable appetite for both money and personnel time.[3] Hence the general manager asks himself: What is software?

### *Software defined*

There are many definitions of software. Because it is a new term in a new industry it is given different meanings in different operational contexts. In 1963 software was considered by Ned Chapin to consist of a body of techniques to make the hardware function effectively. Dr. Chapin described software as the operating knowledge and accumulated experience in the form of aids to the computer user. He defined ten major types of software: operating manuals and guides, programing languages, program-generating routines, utility routines, library routines, diagnostic routines, programming assistance, canned applications, equipment maintenance service, and training[4]—in other words, everything in a computer installation except the computer equipment.

However, this definition of software is too broad for the purpose of this article. Hence definitions by International Business Machines Corporation and Norman L. Enger will be used as benchmarks.

IBM, in *Principles of Programming,* states that software is "all the programming systems required for an effective processing operation, in addition to the hardware of the computer system itself. It includes assemblers, compilers, utility routines, et cetera." Norman L. Enger, in the glossary of *Putting MIS to Work,* defines software as "the totality of programs and routines used to extend the capabilities of computers, such as compilers, assemblers, routines and sub-routines."[5] These similar, yet different, definitions emphasize the communications problem encountered by a general manager trying to grasp the fundamentals of computer software.

### *Software economics*

A company treasurer signs a check for $30,000 in payment for a five-year lease of a proprietary computer program. A computer manufacturer announces availability of his COBOL compiler at extra cost; customers who want it must pay for it. The best known language, FORTRAN, is merely one of approximately 120 higher programing languages. Of this total, nearly 20 are never used or are on obsolete computers; nearly 35 are used very little; about 50 are for use only in specialized areas; and only 15 are widely used.[6]

These are signs of the economic problems and the economic evolu-

[2] J. Richard Sherman, "Toward the Complete Executive—Brainware and the Computer," *Data Processing Magazine,* August, 1969, p. 22.

[3] Daniel W. McElvee and James E. Fernader, *A Software Primer For Managers,* Industrial College of the Armed Forces, Washington, D.C., 1965, pp. 1-2.

[4] Ned Chapin, *An Introduction to Automatic Computers,* D. Van Nostrand Co., 1963, pp. 205-207.

[5] Norman L. Enger, *Putting MIS to Work,* American Management Association, Inc., New York, 1969, p. 235.

[6] Jean E. Sammet, *Programming Languages: History and Fundamentals,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1969, p. vi.

of computer software, states G. W. Armerding. It might be called economic evolution to differentiate it from the economic revolution that has been continually in effect over the past twenty years in computer hardware, that is, speeds have continually increased and prices have continually gone down. But along with this continual hardware economic revolution slower economic evolutionary changes have been occurring within the software area. These changes, while not as conspicuous and sensational as those in hardware, are still important and may be even more important over the next several years.

One of the changes is that the hidden costs of software have been brought into the spotlight. As Mr. Armerding points out, we have all been indoctrinated with the idea that the manufacturers give away their computer software. They say it is "free," and indeed the cost of obtaining the software from the manufacturer, after you have purchased his hardware, is or was zero. But, by any method of evaluation, much of what you paid the manufacturer is for the software, not for the brilliant, neat, colorful units of electronic parts. Managers have become cognizant of these supposedly "hidden" costs and are beginning to make firm motions toward their control or reallocation. Within the ranks of computer managers, we have heard specific recommendations that manufacturers should price all of their software separately, and some manufacturers have taken action. Hence, the customer may now begin to shop and buy only the items of software he needs.[7]

***By any method of evaluation, much of what you have paid the [computer] manufacturer is for the software, not for the brilliant, neat, colorful units of electronic parts.***

WILLIAM E. LINDSAY, commander, Supply Corps, U.S. Navy, received his B.S. from Pennsylvania State University. He was awarded his M.A. by George Washington University and is now working toward his doctorate at American University. Commander Lindsay has published several articles in the **Navy Supply Corps Newsletter**. He was awarded a Navy commendation medal for service aboard ship in Vietnam waters in 1968.

Awareness of the software pricing problem seems to have manifested itself in the middle and late 1960's. In 1967, Richard C. Jones, President, Applied Data Research, Inc., stated that with few exceptions, the dollar cost of preparing application programs in 1967 was the same as in 1957, primarily, he said, because a lack of progress in systems programing has retarded the growth and application of computers since they were invented. Hardware improvements have caused the cost-performance ratio of equipment to improve steadily, but no similar software innovations have been developed to minimize programing time or make the hardware easier to manipulate. Mr. Jones then went on to say that computer manufacturers seem to have had one prime reason for producing software—profit.[8]

In early 1968 Martin A. Goetz, Mr. Jones' vice president at Applied Data Research, Inc., stated that many persons believed that the then current software gap could be traced to an apparent software monopoly that began about fourteen years ago in an innocent manner. In 1955, with the advent of the UNIVAC II and the IBM 705, it was becoming more and more evident that a great number of computer programs were of a very general nature and applicable to many users. Since such programs would aid in computerizing applications, the hardware manufacturer was quick to develop and distribute such programs. These programs not only helped machine sales but also contributed to the belief that the manufacturer assisted the user by providing no-cost aids for programing. While this practice at one

[7] George W. Armerding, *Computer Software: The Evolution Within The Revolution,* Rand Corporation, Santa Monica, California, 1968, pp. 1-14.

[8] Richard C. Jones, "Systems Programming—The Expensive Giveaway," *Data Processing Magazine,* September, 1967, p. 26.

time contributed to the growth of computing, Mr. Goetz believes that it has stifled the most effective use of computers.[9]

In 1968 Dr. Melvin E. Conway, an independent consultant, appeared to be in the minority in expressing the idea that separate pricing of hardware and software was not a black and white argument. He explored the economics of software by organizing his discussion around the following four common confusions: the cost-price confusion, the confusion that software costs as much as hardware, the design-reproduction cost confusion, and the software-support confusion. He arrived at the following conclusions: It appears that software cost does not now contribute a large fraction of the price of a System/360 (as an example); it is probably less than 3.33 per cent, on the average. Software costs are sensitive to economics of scale, however, and attempts to distribute software development among nonmanufacturers will tend both to raise the price to the user and to discourage the manufacturer from undertaking certain products and services. It appears that separation of software pricing and the volume discounting which this implies will help software houses and large users but will discriminate against smaller users, smaller manufacturers, and those manufacturers who concentrate on serving the Federal Government.[10]

At this time, as the actual separation of hardware and software pricing is accelerating, the entire cost/price structure of software is in a state of flux. The intelligent and aggressive general manager must be aware of what is happening daily in this area of concern, make his decisions on the latest information, and resist making any long-term contractual arrangements for software.

Way back when in the history of computers, say, ten years ago, the byword of the computer programer seemed to be efficiency, according to Mr. Armerding. Only incompetents would use such a thing as a trace program. Interpretive programs were used only with careful supervision because they did not employ the computer efficiently.

Now, however, we are in the midst of the hardware revolution, and the software world is adopting a more enlightened attitude toward inefficient machine usage. Consider time sharing as an example. Few eyebrows are raised when it is reported that a general purpose time sharing system uses 50 per cent of the available computer time performing its various overhead operations. Or consider interpretive programs; it is found that even the installed compiler, one of the most frequently used programs, runs interpretively, making the computer act like something it is not. Inefficient? No. The experts claim the compiler is more than worth the time it takes. Operating systems, loaders, editing programs, and many other overhead functions take a large share of the central processing unit cycles away from the problem program. The new generation of programers finds nothing wrong with this mode of operation even though it makes the old-timers cringe. To the new man the benefits exceed the costs. It is true that everybody would like to obtain as much useful work out of the computer as possible, but, everything considered, we are paying less, a great deal less, for each useful answer than we paid in the past. Increased hardware performance per hardware dollar more than compensates for the loss of the ever-increasing portion of the machine's power absorbed by software.[11]

In fact, Arthur Nesse, of the Ford Motor Company, projected figures for hardware versus software shipments that indicate that this trend can be expected to continue. According to Mr. Nesse, in the 1960's hardware represented approximately 60 per cent of the value of computer shipments. By 1975, Mr. Nesse quotes the Standford Research Institute as predicting, the value of the hardware component in computer shipments will decline from 60 per cent to about 30 per cent or 40 per cent, and the value of software will grow to the complementary 70 per cent or 60 per cent.[12]

As long as the hardware designers keep reducing the cost of computers and keep raising their performance, users will perhaps demand, and for certain tolerate, more and more "overhead" software to make things easier for the programer and his program. Only when the designers of computers have wrung the last drop of power out of the circuits and the manufacturers have reduced their production costs to their reasonable limits may we expect the software builders to begin to be seriously upset about software overhead rates. That day appears far in the future.[13]

### *Programs for sale*

In the very early days of the computer industry, the machines were installed with virtually no software. Users programed in machine language and had to develop their own software to supplement what little the manufacturer did supply. Rapidly it became evident that there were programs that almost all users needed, since many users were individually writing almost identical programs. Informal and later formal user groups were established to share and exchange programs of mutual interest. The concept of developing programs of general use and hence saving people and money that would other-

[9] Martin A. Goetz, "Proprietary Programs —Can They Break the Software Monopoly?," *Data Processing Magazine*, January, 1968, pp. 48-49.

[10] Melvin E. Conway, "On the Economics of the Software Market," *Datamation*, October, 1968, p. 31.

[11] Armerding, *op. cit.*, p. 6.

[12] Arthur C. Nesse, "A User Looks at Software," *Datamation*, October, 1968, p. 49.

[13] Armerding, *op. cit.*, pp. 5-6.

wise be wasted in "reinventing the wheel" had its beginnings in these early cooperative efforts. These shared programs were among the first software.[14]

The software economic evolution is also at work here. The early spirit of "together we stand, divided we fall" has been all but lost. Now any program worth the cards it is punched on is being offered, not free, but for a price. The computer magazines are full of ads. In the want-ad section of a computing newspaper, an individual programer asks $150 for a copy of his improved, high-speed sort program.[15] As a further example, in the March, 1970, issue of *Datamation* there were seven pages of ads for software containing twenty-nine advertisements.[16]

***Every programer in the country is a potential software seller. All he requires is very little capital, coding paper, pencils, and a few hours of weekend or evening time.***

The early software entrepreneurs found the market quite rough; a few early programs offered for sale did not make enough sales to cover expenses. The market now appears to be mixed. Managers, who at first were revolted at the idea of having to pay for something they had traditionally received "free," recognized the inevitable. Now managers are spending substantial sums for the privilege of using proprietary program packages. However, reckless competition is almost sure to come. Every programer in the country is a potential software seller. All he requires is very little capital, coding paper, pencils, and a few hours of weekend or evening time. Countless time sharing installations will be pleased to sell him machine time to debug and test his programs.

Now that many shots are being heard, this part of the software economic evolution may well become a full-scale revolution. No one should be amazed by the further avalanche of proprietary programs on the market. It may not be too long before the day arrives when all software will be sold in an environment of true competition.[17]

### *A forecast*

With the advent of separate pricing of software by some manufacturers and of proprietary programs offered for sale in great numbers, the time is ripe for significant economic changes within the realm of computer software.

At the present time most of the programs offered for sale are oriented toward applications. Therefore, they seldom compete directly with software offered by the computer manufacturer. But enterprising programers have begun to compete with the manufacturers and will continue to do so. The market may see improved versions of "free" software, or software that will replace the "free" software, or programs that supplement the standard software or make it easier for users to approach. This, of course, will lead to a highly competitive market. The manufacturers appear to have the upper hand. They have great freedom to adjust the price of their software, from gratis to profit-making. But as high-performance substitutes appear on the market, the pressure will be on the manufacturers either to improve their performance or to adjust their pricing.[18]

### *Software standards*

Standards for software are rather like control for weather. A lot of talking has been done, but few results have been attained—and for the same underlying reason: Good software and good weather are not the same to each person.

It is true that one of the key factors in the definition and use of software is the role played by standardization. Another basic purpose of standardizing is to achieve compatibility, which in turn reduces personnel and documentation cost.

[14] Robert V. Head and Evan F. Linick, "Software Package Acquisition," *Datamation,* October, 1968, p. 22.
[15] Armerding, *op. cit.,* p. 8.
[16] *Datamation,* March, 1970, p. 215.
[17] Armerding, *op. cit.,* pp. 8-9.
[18] *Ibid.*

Standardization also assists in converting to new computers. Despite all of these advantages, there has been only limited success in standardizing software under the American National Standards Institute (ANSI). Two items that have been standardized are the higher-level languages FORTRAN and COBOL.[19] Certainly, any software could be standardized if the need were acute; consider, for example, the work being done in machine tool control.

There are currently three major forces in standardization: ANSI, the European Computer Manufacturers Association (ECMA), and the National Bureau of Standards (NBS), within the United States Department of Commerce. ANSI is a federation of nearly 150 trade associations and professional societies and more than 2,000 member companies. It is a privately supported organization acting as the national clearing house and coordinating agency for voluntary standards in the United States. The word voluntary is important. There is no force of law behind ANSI's standards, nor is there even an implied commitment on the part of those responsible for developing a standard that they will later support or use it.[20]

The European Computer Manufacturers Association was formed in 1960 and is largely, as the name implies, a manufacturers' association. This group's significance in American standards work is twofold. First, it acts as a challenge to United States standards work. Second, the ECMA acts as a second court of appeals for American standards.[21]

On March 11, 1968, the Federal Government approved the first Federal automatic data processing standards. This was the first step in an intensified effort by the Federal Government to strengthen the control of Federal computer activities. Three Federal agencies are responsible for the development of data processing standards: the General Services Administration, the Bureau of the Budget, and the Department of Commerce (National Bureau of Standards). The NBS has been authorized to make recommendations to the President relating to uniformity of Federal automatic data processing, and it also has responsibility for the promotion of voluntary commercial EDP standards.[22]

### *Problems of standardization*

There are five major problems in approaching standardization: conceptual problems, technical problems, procedural problems, time, and expense.

The first conceptual problem is one of timing: When should the standardization of software take place? Without careful consideration, standardization is likely to come too soon or too late. If it is too soon, there is a risk of standardizing things that are not really very good. On the other hand, if standardization is delayed too long, then innumerable variations have developed, many of them representing only minor differences, which means a number of vested interests that are reluctant to accept a standard that diverges from their particular version.

A second conceptual problem is the risk of smothering progress. In some manner the standardization process must avoid preventing or eliminating technical progress. This is difficult because there is no easy way of coping with bright new ideas if they come up after the standard is established, or even while it is in the process of being established.

The technical problem of software standardization is one of definition. We do not yet understand how to define software with rigor.[23]

***Standards for software are rather like control for weather. A lot of talking has been done, but few results have been attained and for the same basic reason: Good software and good weather are not the same to each person.***

[19] Sammet, *op. cit.*, pp. 43-47.
[20] Paul B. Goodstat, "Standards in Data Processing," *Data Processing Magazine*, March, 1967, pp. 22-25.
[21] Don Crayford, "A Future for ECMA?," *Datamation*, September, 1969, pp. 43-44.
[22] Enger, *op. cit.*, pp. 195-196.
[23] Sammet, *op. cit.*, p. 44.

No completely formal method exists for the definition either among or within the software classifications of operating systems, programing languages, time sharing software, data management, and applications software.

The procedural problems in establishing standards are enormous, but they are unavoidable. Standardization must be undertaken cautiously to prevent the issuance of undesirable standards. (Undesirable merely means not acceptable to virtually all the user groups to whom the standard will apply.) The delays in the establishment of a standard, often three and one-half to five and one-half years (the time problem), are caused by the complexity of the procedures, which have been designed to protect the rights of those involved.[24] This often causes difficulty for those groups that are at a stage in their technological or manufacturing development where they are eager to implement a standard, which does not yet exist officially and still may be altered.[25]

The final problem in approaching standardization is its tremendous expense. With the amount of work involved in carrying a standard from working group meetings to a proposed standard, through a series of committees, and finally to ANSI approval and the printing as an ANSI Standard, it should be clear that a lot of expense is involved. The heavy cost must be borne by trade associations, manufacturers, consumer groups, general interest groups, and even interested individuals in some cases. In addition, there is the enormous expense of converting software in use to meet an agreed-upon standard.[26] Those dedicated individuals who are responsible for what progress has been made must be admired for their good sense and their tenacity against what must appear as enormous odds.

General David Sarnoff, in his keynote address to the 1964 Fall Joint Computer Conference, said, "Standards can be established which, if planned with thought and foresight, can guide us in the future, linking our separate efforts and facilitating the common evolution of our industry. Such standards are indispensable to continued progress." Unfortunately, it appears that General Sarnoff's hopes will be a long time coming true in computer software.

***Potential legal problems involving software are as infinite in number as the possible advances and applications of software. Unfortunately, many of these problems have gone unrecognized because of communication barriers. Lawyers are not aware of software developments and their practical applications and hence do not foresee trouble areas.***

### *Software legalities*

"The Software Case Is Settled," reads a headline from *Data Processing Magazine*, September, 1969. The article continues, "Now patentable after many months of controversy and dispute, it becomes even more significant in the light of recent separation from the hardware market. The U.S. Court of Customs and Patent Appeals ruled favorably on the appeal of Charles D. Prater and James E. Wei on the grounds that once a digital computer is properly programed, it becomes a special purpose computer (a specific electrical circuit). The Court rejected the Patent Office contention that programing a general purpose digital computer is 'obvious.' "[27]

What does this news mean to the general manager? How will this legal decision on software affect him, and are there other legal areas such as software contracts, taxes, and employee contracts that he should be aware of? (The answer to the last question is a firm yes.)

Potential legal problems involving software are as infinite in number as the possible advances and applications of software. Unfortunately, many of these problems have gone unrecognized because of communication barriers. Lawyers are not aware of software develop-

[24] Gilbert E. Jones, "The Impact of Standards," *Computers and Automation*, May, 1969, pp. 38-39.

[25] Sammet, *op. cit.*, p. 46.

[26] E. Stuart Fergusson, "USASI and Formal Standards Activities," *Data Processing Magazine*, April, 1967, pp. 43-44.

[27] "The Month That Was," *Data Processing Magazine*, September, 1969, p. 8.

ments and their practical application, including the law of trade tions and hence do not foresee trouble areas. Management and computer technical personnel, left on their own, may forge ahead with new ideas without an understanding of the complex legal consequences. Even those who are sensitive to these problems may be reluctant to seek legal advice at every juncture, and lawyers may be correspondingly uneasy about offering opinions in such a complicated and uncharted area.[28]

To make matters more complex, we have not so much an undefined product as a product whose definition keeps changing. Software may be viewed from several levels. It can be simply the program, or it can be the program plus the research effort that has been expended on the total study of the problem. Think, for example, of all the programing for a time sharing system. It is all software. But the individualized customer programs, each a subset, are marketable commodities in and of themselves, as are the executive routine and, maybe, the documentation. Then there are the constantly changing relationships among hardware and software. Again using time sharing as an example, as the problems of the security of data have become more important, what used to be software is now being built into hardware, and the term firmware has been used to define the developing concept of a hybrid personality.[29]

Even with the recent patent decision, there remain two distinct arenas for the discussion of the protection of the program developer's proprietary interests in the software against unauthorized use. The first arena is that of the law of intellectual property, primarily expressed by statute in patent, copyright, and trademark laws; the other arena is that of the common law, including the law of trade secrets, unfair competition, and contracts.[30]

Now that the patent decision has been made, however, where does this problem stand? Allen W. Puckett, an attorney with McKinsey & Co., Inc., predicted in 1967 that even if patents were granted their use would be severely restricted. His prediction has been somewhat substantiated by the infrequent use of the copyright in registration of computer programs. In May, 1964, the Copyright Office decided that it would allow the registration of computer programs. By mid-1967, within a period of more than three years, only about a hundred computer programs had been registered.[31]

Mr. Puckett offers four reasons why patents will not be used to any great extent: First, patents are very expensive. To get a patent requires not only significant legal fees but also lost programer time, and programer time is a critical commodity that industry is seeking to conserve.

Second, obtaining patents is very time-consuming. On the average, the lapse from the time an application is originally submitted to the time a patent is approved is about three years, and then the patentee has merely a "license to litigate." If the patent then is litigated, protection is even further away. In view of the past rate of development of the industry it is likely that the program will be obsolete by then.

Third, a program patent is risky. An obvious risk is that large expenditures may have to be made defending the patent. If the patent is struck down by the courts, the opportunity for other types of pro-

***To make matters more complex, we have not so much an undefined product as a product whose definition keeps changing. Software . . . can be simply the program, or it can be the program plus the research effort that has been expended on the total study of the problem.***

[28] John F. Banzhauf, "When Your Computer Needs a Lawyer," *Communications of the ACM*, August, 1968, pp. 543-544.
[29] Robert B. Bigelow, "Legal Aspects of Proprietary Software," *Datamation*, October, 1968, pp. 32-34.
[30] The reader is referred to both Allen W. Puckett, "Protecting Computer Programs," *Datamation*, November, 1967, pp. 55-60, and Robert P. Bigelow, "Legal Aspects of Proprietary Software," *Datamation*, October, 1968, pp. 32-39, for a detailed discussion of the total relationship of software to statutory and common law.
[31] Bigelow, *op. cit.*

tection, such as copyright, trademark, trade secret, unfair competition, and contract, may well have vanished. For instance, the program would no longer be a "secret," protectable against unfair competition.

Finally, in practice, program patents would probably be unenforceable for all but major corporations because of the prohibitive litigation expenses and the extreme difficulty in detecting patent infringements. To obtain a patent, an inventor must file a description of the invention with the patent office. A complete copy of that description may be obtained from the Patent Office in return for a small sum. Since creating the program from a detailed description would be inexpensive and since programs would be duplicated for use rather than for resale, it would be almost impossible for a patent holder to track down the clandestine users of a patented program.[32]

What is the long-range solution to this problem? It appears that the best solution would be program protection provided explicitly by a new act of Congress.

### *Short-range protection*

According to Robert P. Bigelow, the best current protection of purchased software seems to be a contract between the manager and the supplier. There are several important items that must be covered in any contract for software. Where a program is to be developed by an application company for the user, the contract should specify, in addition to such important items as the time schedule and the price, the purpose of the program, the documentation required, the on-site assistance to be rendered by the software house, and, above all, the ownership rights in the program.

The standards of performance the software is to meet must be spelled out in detail for both the user and the developer, and the developer should be required to correct all errors found. To date, the literature shows, there have been no cases litigated on failure to meet contract specifications on software, but there have been two cases dealing with the problem of a hardware supplier's failure to fulfill the terms of its contract. In one, decided in the state of New York, the Federal Reserve Board spelled out in great detail what it was to receive. The hardware supplier was unable to produce; the damages were over a quarter of a million dollars (U.S. v. Wegematic Corporation, 360F2d674).

### *Liability problems*

The manager may be liable to someone who is hurt, without that person's having to prove negligence, particularly if the program is of the process control type where a failure to meet specifications could have dangerous results. This would be similar to the cases involving exploding soda pop bottles or cars that lose wheels. There is a distinct trend in the courts to dispense with the requirement that a person injured under such circumstances prove that the defendant was negligent. Probably we can expect to see this thinking applied when computer programs are operating and something blows up. The contract should cover this liability, and both parties should attempt to obtain insurance coverage against such an event.

The contract should also cover the developer's liability for the infringement of the rights of others. While the developer's own rights in software against unauthorized use are becoming clearer, it is possible that in developing the program he might infringe a copyright or a trade secret, particularly if, in developing the program, he used someone who had been under a restrictive agreement with another computer-oriented firm.

Debugging and testing times should also be included in the contract. In a recent issue of *Computerworld,* a user was quoted, "My fight with the manufacturers is over their software contracts. You have to start paying while you are still testing, and you can't cancel for three months. I don't see why we should pay for the privilege of testing software and advising them what is wrong with it."[33]

Particularly important in contracts for software is provision for penalties. The history of software has been one of dilatoriness. The State of California, as of 1968, is putting a penalty clause into all of its contracts for software. It has been reported that IBM has signed a contract with the State of California to provide software which must "show substantial conformance to the manufacturer's specifications," with penalties for failure to meet such specifications on time. This type of contract clause may become quite common, not only in government procurement but also in acquisition by sophisticated managers. And into the bargaining equation will go the normally heavy economic weight of the user as compared to the normally light weight of the software house.

The contract may also include clauses specifying whether the software is or is not for one user only; the user's rights to improvements made by the software developer; the user's rights to make modifications; and perhaps the developer's rights to improvements made by the user. On the other hand, when software is leased or rented, the manager may have to sign a contract which carefully spells out the reservation of rights.

These are some of the matters which should be considered in contracts for proprietary software. The best protection for the manager, however, is to deal with an honest man and give him a square deal.[34]

Even in the software area, taxes

[32] Puckett, *op. cit.*

[33] Phyllis Higgins, "Users Resisting IBM SE Contract Selling," *Computerworld,* February 11, 1970, pp. 1-2.

[34] Bigelow, *op. cit.*, pp. 37-38.

must be considered, says Mr. Bigelow. There is a good argument that developmental costs for software should be treated as an expense item. Accountants who have looked into the matter have defined software to include the justification study, the feasibility study, the systems work, and the training of personnel as well as the actual programing—in other words, everything related to the installation of a computer system except the hardware costs. Obviously, these costs can be considerably higher than the actual hardware outlays, especially if the hardware is rented. From the manager's point of view, the price or rental of software is but the visible portion of his software costs. There are some useful precedents going back as far as 1925, including outlays for efficiency systems, management surveys, revisions of accounting systems, and so forth, to indicate that the proper tax treatment, at least from the manager's point of view, may be to take all software costs as an expense of doing business in the year in which they were incurred.

It has also been suggested that software development costs should be treated in the same manner as research and experimental expenditures. Certain expenditures of this nature can be handled either as capital or expense items to be amortised over a period of not less than five years. Once the choice is made, you have to stick with it.

From another point of view hardware is tangible personal property which has a useful life of more than one year. It is depreciable, tangible personal property to which all the depreciation rules for tax purposes apply and for which an investment tax credit may be taken. The interdependence of hardware and software and the growing problem of deciding where the line is between the two give weight to the argument that software development costs should be treated the same as hardware for tax purposes. The investment credit is available for depreciable tangible personal property which is used as an integral part of a manufacturing operation. "Integral part" is defined to include cases where the property is "used directly in the activity and is essential to the completeness of the activity." A computer program which is used for process control would seem to fit this requirement.[35] In sum, tax regulations are important, and the proper handling of software cost may result in significant savings.

### *Contracts with programers*

A final legal aspect of software that should be considered is the relationship between the company and its programers. Mr. Bigelow has set forth the following concepts in this field:

When a product is developed by a team the individual employee has comparatively few rights. But what about the situation where the product is developed by a full-time employee on his own time? One company in Boston had a problem of just this sort. The product in question was an exceedingly valuable program which the company, which is not in the program development business, nevertheless hoped to be able to peddle. But the employee, who put in a great deal of his own time, also wanted to make some money. The only clear answer to such a problem and problems like it in the future is a clearly written employee contract covering such questions. Such a contract should also cover relationships between the employer and the employee after the employee leaves the company.

To cover relationships during employment, the contract might well include the following: (1) an agreement to disclose all intellectual accomplishments of interest to the company, whether made on company time or on the employee's time, if the discovery is capable of being used by the company, (2) an agreement to execute such assignments and other papers as the

***A contract for software might include clauses specifying whether the software is or is not for one user only; the user's rights to improvements made by the software developer; the user's rights to make modifications; and perhaps the developer's rights to improvements made by the user.***

[35] *Ibid.*, p. 38.

company may request to give it appropriate rights in such discovery, together with a representation that there are no such discoveries at the present time; this latter item can be very useful in avoiding arguments and litigation later.

To protect the company's property rights, the contract may provide that the employee will keep confidential information secret forever, whether related to the company, its programs, or its products. He should also agree that if he leaves he will not, without written consent, take with him processes, formulae, and so forth relating to the company's operations or its experiments.

Of equal importance are contractual arrangements after the employment is over. Most agreements of this type which have come before the courts have been agreements not to establish a business, such as a restaurant, within a certain geographical area. In the software field, geography is irrelevant. If a manager wants a noncompetitive agreement, it is suggested that it be put on a time basis. As an illustration, when you employ a person, get him to agree that for three months after he leaves he will not engage in any activity that competes with any business in which the company is engaged at the time he leaves and that during a full year after he leaves he won't compete directly with the company in any such business. Reasonable time limits will be upheld but the courts will not deprive a man of his livelihood forever. Even without a contract, if it can be clearly shown that the former employee made unauthorized use of information which he had received from his employer, he can be enjoined from using it and made to pay damages.

One important item the manager may wish to include in a programer's contract is an agreement that he will, after termination, upon payment of an amount specified in the contract, return to work for the original company for the finite purpose of updating programs on which he worked. With the difficulty in updating programs, such a clause might avoid the risks of undocumented changes made by short-term employees.[36]

### *Summary*

This article has discussed software, from a general manager's point of view, in three specific areas, economics, standards, and legalities. In all these areas the overall impression is one of great flux and change, monthly if not weekly. Hence, the general manager must make it an explicit work habit to keep himself totally and daily informed of the changes and the proposed changes in these important software areas.

---

[36] *Ibid.*, pp. 38-39.

---

## Bibliography


Ambrose, John, "The Tale of Crazy Freddie," *Computers and Automation,* February, 1970, pp. 14-15.

Carlson, Charles E., and Philip C. Semprevivo, "Multi-Programing Computer Evaluation and Management," *Management Services,* September-October, 1968, pp. 39-43.

Dansiger, Sheldon J., "Proprietary Protection of Computer Programs," *Computers and Automation,* February, 1968, p. 32.

Dorn, Philip H., "Obsolescence: Systems Software," *Datamation,* January, 1969, pp. 25-36.

Fergusson, E. Stuart, "USASI and Formal Standards Activities," *Data Processing Magazine,* April, 1967, pp. 42-46.

Fisher, F. Peter, and George F. Swindle, *Computer Programming Systems,* Holt, Rinehart and Winston, Inc., New York, 1964.

Galler, Bernard A., *The Language of Computers,* McGraw-Hill Book Company, New York, 1962.

Glans, Thomas B.; Burton Grad; David Holstein; William E. Myers; and Richard N. Schmidt, *Management Systems,* Holt, Rinehart and Winston, Inc., New York, 1968.

Goodstat, Paul B., "USASCIT, What's It All About?," *Data Processing Magazine,* June, 1967, pp. 20-24.

Kendall, M. G., "Software Costs—A Plea for Separate Pricing," *Data Processing,* March-April, 1969, p. 117.

Ledley, Robert S., *Programming and Utilizing Computers,* McGraw-Hill Book Company, New York, 1962.

Martin, James, *Design of Real-Time Computer Systems,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1967.

Moore, Michael R., "Pitfalls in Planning an EDP Installation," *Management Services,* September-October, 1968, pp. 25-32.

"New Software," *Datamation,* September, 1969, pp. 221-227.

Optner, Stanford L., *Systems Analysis for Business Management,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1960.

Reynolds, Carl H., "Notes on Estimating, and Other Science Fiction," *Data Processing Magazine,* June, 1967, pp. 44-45.

Reynolds, Carl H., "Software Standards," *Data Processing Magazine,* March, 1967, pp. 26-28.

Reynolds, Carl H., "The Research Institution and Data Processing," *Data Processing Magazine,* April, 1967, pp. 60-61.

Rosen, Saul, *Programming Systems and Languages,* McGraw-Hill Book Co., New York, 1967, p. 23.

Scharf, Tom, "Management and the New Software," *Datamation,* April, 1968, pp. 52-59.

"Separate Hardware/Software Pricing?," *Datamation,* June, 1968, pp. 72-78.

Silber, Howard A., "A Hypothetical Interview Between the President of a Computer Software Company and a Patent Attorney Specializing in Protection of Computer Programs," *Computers and Automation,* February, 1970, pp. 14-15.

Silber, Howard A., and John Ambrose, "IBM, the Patent Office, and the Small Software Company: The Emergence of an Industry," *Computers and Automation,* February, 1970, pp. 14-15.

"The Problems of Packaged Programs: The American Management Association Conference Report," *Datamation,* April, 1968, pp. 75-79.

"The Software Explosion," *Business Automation,* September, 1968, pp. 24-29.

"The Year 2000," *Computer Digest,* April, 1969, p. 14.

U.S. Congress, House Committee on Government Operations, *Data Processing Management In The Federal Government,* hearings before a subcommittee of the Committee on Government Operations, House of Representatives, 90th Congress, 1st Session, 1967.

Weinberg, Gerald M., *PL/1 Programming Primer,* McGraw-Hill Book Company, New York, 1966.

"What To Do Until The Software Package Guru Comes," Editorial, *Datamation,* October, 1968, p. 21.

Wilker, M. V., and D. F. Hartley, "The Management System—A New Species of Software?," *Datamation,* September, 1969, pp. 73-75.