TRAINING REQUIREMENTS FOR COMPUTER PROGRAMMERS IN RELATION TO SYSTEM DEVELOPMENT PHASES

PERRY E. ROSOVE
System Development Corporation

INTRODUCTION

The purpose of this paper is to suggest that the training of computer programmers might be improved by developing training requirements and course curricula through analyses of the attributes of typical information system development phases. This approach differs from, but would complement, the existing method of deriving training requirements from analyses of the tasks and skills associated with job categories.

At the present time the training of systems personnel is derived from distinctions made among such job categories as keypunch operator, computer operator, computer programmer, systems analyst (sometimes referred to as systems designer) and supervisor or manager of data processing. Some of these job categories are also thought of as constituting a career ladder. Typically, an individual's career may begin as a computer programmer and progress to the higher positions of systems analyst and, ultimately, manager of a data processing group or service facility. In this paper we shall limit our concern to this career sequence from programmer to manager of data processing, since it constitutes the most difficult area for the training designer or educator. The tasks performed by operators (keypunch, tabulator, computer, verifier, etc.) are relatively well defined and widely accepted by contrast.

There is much dissatisfaction with the existing training provided for computer programmers. The general situation has not improved notably in recent years. In 1965, a report on electronic data processing published by the American Management Association stated that the "lack of trained EDP personnel is one of the foremost problems today. . . . Too few data-processing courses are being given, and the lack of agreement about qualifications for a programmer or a systems man does little to alleviate the shortage."[1]

As recently as September, 1968, a University of Colorado study of the impact of automated data processing on educational institutions concluded that "data processing personnel need to be oriented to the total systems approach in business and the schools are not meeting the needs in training personnel for the many job opportunities in data processing in business. Much more effort needs to be exerted by school boards, school administrators, teachers and state supervisors to inaugurate curriculums in data processing and to update the programs currently in existence."[2] Uncertainty over the kind of training computer programmers ought to have has reached the point of rather acrimonious dispute be-

tween industrial employers of programmers and the operators of electronic data processing schools. The employers complain that the schools provide machine-oriented training when they should be offering systems-oriented training. The schools, on the other hand, criticize the employers for insisting that a programmer trainee should hold a college degree and that they have failed to identify the specific training they require of programmers.[3]

That there do exist inadequacies in the training of computer programmers has been confirmed by the writer during the preparation of a recently completed history and evaluation of the development of a large-scale, computerized air defense system.[4] One of the more interesting conclusions reached in this study was that many of the programmers who participated in the project lacked a systems viewpoint and were particularly weak in the area of systems engineering.

Despite the plethora of words in recent years about the importance of the systems point of view, it is noteworthy that this perspective is conspicuously absent in the actual work of programmers and in the training they receive. A review of the training curricula for programmers reveals most typically courses on specific types of computers, types of higher order languages and types of applications. Variations in curricular content are geared to job categories rather than to analyses of what may have to be done during specific phases of a system's development. One cannot find a curriculum which takes two essential dimensions into consideration: (1) the job level the programmer has reached in his career; and (2) the phase of the system development effort in which he may or might be engaged. The widespread neglect of this latter dimension may be attributed to the notion that once a programmer has been trained to design and code computer programs, he can learn to operate effectively in *any* system development phase as part of his on-the-job experience. However, it is suggested that this approach to programmer training is no longer adequate. We now know enough about the general phases of system development so that we can begin to relate programmer training not only to job categories but also to system development phases. Table 1 presents a matrix which systematically relates programmer tasks and functions and the training requirements derived therefrom to system development phases. The phases which appear in the left-hand column of the table are taken from a previous work by the writer in which five phases were identified: requirements, design, production, installation and operations.[5] The last phase, operations, is not discussed in this paper since it has little relevance to our purpose.

In the balance of this paper, we shall review in sequence the typical tasks and functions performed by the systems man or woman during the system development phases and also suggest the training requirements implied by those tasks and functions.

It is not possible or necessary to review here all of the training requirements which are listed in the right-hand column of Table 1. In the following pages we have selected for discussion only those aspects of training which are believed to be the most neglected in existing formal courses of instruction for systems people and which might be regarded as the most controversial.

Table 1. Relationship of Training Requirements to System Development Phases

Development Phases	Tasks and Functions	Training Requirements
System Requirements Phase	Interaction with top managers and administrators; determination of user objectives; analysis of user organization; analysis of user system requirements; evaluation of alternative system concepts; development of long-range system acquisition plan; analysis of future requirements; preparation and writing of system specifications; "selling" system plans to top management in user organization.	Ability to interact with top management; interviewing techniques; organization theory; cultural differences; systems engineering; systems analysis; cost benefit analysis; forecasting methods; writing ability; salesmanship; oral presentation.
System Design Phase	Translate system specs into system design requirements; allocate operational functions among men, equipment and computer programs; preparation and writing of system design specs; determine requirements for personnel, training and system evaluation; development of a test plan for system qualification and acceptance.	Systems engineering; systems analysis; equipment capabilities and constraints; simulation and modeling; task analysis and human engineering; testing methods and techniques; organization and job design; training methods and techniques; training methods and techniques writing ability.
Production Phase	Translate system design specs into computer program design specs; design computer programs; code programs; prepare flow diagrams and program descriptions; test and debug programs; test groups of related programs; test total program system; document computer programs for system maintenance, configuration management, and hand over to users.	Computer program design; flow charting; equipment capabilities and constraints; personnel capabilities and constraints; testing methods and procedures; design of test tools; computer program and testing documentation; writing of user guides and manuals; training aids and materials; configuration management procedures and documentation.
Installation Phase	Prepare installation plan; provide system orientation and training for users; install system in operational environment; test, debug and evaluate system; recommend design changes; configuration management.	Human relations; capability to provide instruction and conduct training at all levels of user organization; testing and evaluation methods and techniques; configuration management procedures and documentation; scientific method and statistics.

SYSTEM REQUIREMENTS PHASE

An examination of the training requirements cell associated with the requirements phase (see Table 1) suggests some of the inadequacies in current training programs for computer programmers. In terms of the systems man's career ladder, the tasks and functions required in this phase are normally expected of the systems analyst and the manager of data processing. Where do persons in these job categories receive the training to perform these tasks and functions effectively? Presumably they do so as programmers through on-the-job experience in a variety of system development projects and by supplementing this experience with available company-provided, in-house training (in systems engineering, systems analysis, etc) or by taking extension courses in a local college or university (if one is nearby). Unfortunately, this experience and training are not always acquired.

What formal training does the average computer programmer receive to prepare him for the day when he assumes responsibility for determining the system requirements for a user who is the head of a large military command and control system, a large university, a major industrial corporation or a government agency or bureau? Does he come equipped with skills in the area of human relations? Does he know how to interview top managers and administrators about their goals and objectives, their decision-making criteria and their organizational style? Has he been prepared to sell his system ideas, once he formulates them, to these same top managers who may be skeptical of what electronic data processing can do for their organization? The answers to these questions, of course, are either negative or difficult to determine.

An official of a commission of the United States government recently stated at a luncheon seminar to which the writer was invited that the basic problem faced by his organization in developing a computerized information system was not technical but organizational. The knowledge to solve technical problems is available, he maintained. But he could not get his various department heads to agree on standardized data formats and records. This was the area in which he needed professional help which his systems men apparently could not provide.

The systems analyst should be trained at some point in his educational career to anticipate that in large-scale organizations there is a spectrum of structural arrangements from a highly centralized bureaucracy at one extreme to a loosely structured, decentralized organization at the opposite extreme. One may, for example, contrast the type of centralized organization represented by a military command and control system with the decentralized organization represented by a state university. Business enterprises may vary in structure anywhere along this spectrum. The determination of user requirements poses many different problems for the systems man depending upon where the user's organization falls along this spectrum. A major need of the analyst, for example, is to find a key authoritative spokesman from whom he can obtain legitimate user requirements. In a centralized organization this may be relatively easy, while in a decentralized organization extremely difficult. In an organization characterized by autonomous departments the systems man may find himself in the awkward position of attempting to establish standardized procedures, data formats, machine outputs and reports for department heads who insist on their own unique needs in these areas. Is the analyst equipped by his training and education to cope with

problems of this nature? Apparently not, according to the government official referred to above.

Where shall the systems group be located in the user's organization? One study shows the location of the systems group at any one of five different levels below the level of top management.[6] This same study also found that these locations frequently changed as management gained experience with the new system. At the present time the decision of where to locate the systems group is usually made by the user's top management on the basis of largely irrelevant criteria. Should systems men participate in this type of decision? We believe they should, since this is an aspect of the total system design activity. But where does the systems man obtain the necessary training to be able to make such a decision? Certainly he does not get it in current courses of instruction for computer programmers.

It might seem farfetched to suggest that programmers who may eventually become systems analysts ought to receive some training in cultural anthropology as well as organization theory. But the phenomenon of "culture shock" is not an uncommon experience for systems people suddenly immersed in the military subculture, in a federal bureaucracy, in the cloistered halls of academia or in a foreign country.

The following event actually occurred (circa 1960), although in the light of recent history it is increasingly hard to believe. During the system requirements phase in the development of a large-scale, computerized military command and control system, a systems man doing an organizational analysis was required to visit the command headquarters, a very security conscious facility. This particular analyst was distinguished by his very black, very bushy beard. He was observed by the military commander in the command headquarters facility. The commander's reaction to the analyst's appearance was to forbid him any further access to the headquarters facility despite the fact that this made the analyst's work impossible to conduct. Later that evening the writer found himself attempting to placate and console a very distraught analyst who simply could not understand the military commander's attitude towards black and bushy beards. The writer had to explain that in the military sub-culture heavy, black beards were symbolic of the enemy and the well-shaved face a component of the class A uniform. He capitulated against his will and shaved the beard off. Some sensitivity to sub-cultural differences might have saved him considerable anguish.

While this story may seem more amusing than didactic, the same point may be brought out more clearly by noting that many programmers and systems analysts, trained in the United States, are working on management information systems in Europe and on various types of system development projects in such exotic countries as Vietnam and Thailand. In addition to the language barrier, which is formidable enough, the systems man dealing directly with the user should be sensitive to cultural differences which may have a bearing on his work. In his study of European executives, David Granick reports that modern industry in four major countries operates within a social structure which antedates modern industrialization.[7] The basic theme of the book is that there are differences "between national concepts as to the role of management in industrial firms."[8] Thus, in England, for example, the "amateur theory of management" is dominant. Study of the classics and the arts is considered to be most appropriate for the future

top manager. Management is regarded as an art, and there are no accepted "principles of administration." Since management is regarded as an art and not a science, there is little interest in managerial training and professional conferences. A reading of Granick's description of the typical English manager suggests that the problem of obtaining system requirements information (organizational objectives, decision-making criteria, etc.) which the systems analyst finds difficult and frustrating in the United States would be even more difficult and frustrating in England. In France, top managers are drawn from a handful of engineering schools and the attitudes of French managers are quite different from their English counterparts and so on. It would help the analyst if he had some forewarning of such cultural differences. Needless to say, he receives no such training at the present time.

A major feature which distinguishes information systems from other types of systems is that they are experimental or evolutionary in nature. They are continuously subject to change due to changes in operational requirements, additional applications, the introduction of more sophisticated equipment, etc. Military systems are characterized by a series of new "models" or "versions," each new model or version built upon the foundations of earlier ones. In business systems there is rather typically an enlargement of the scope of the system. One begins with an initial application, such as payroll, and gradually incorporates additional functions until what is called a "total integrated system" is achieved. This experimental or evolutionary nature of information systems implies that during the requirements phase the systems man needs to be able to forecast future user system requirements and to be able to prepare long-range plans for the addition of new capabilities to the initial system. Forecasting and long-range planning are specialized skills and include a variety of techniques. Training in these skills and techniques should be provided to systems men on a routine, formal basis. This is not the case.

SYSTEM DESIGN PHASE

During this phase of the system development effort, the systems man is concerned with translating the system specifications written during the previous phase into design specifications which will be used as the basis for detailed computer program design. The total system will be composed typically of a number of basic segments: the computer(s) and peripheral (input/output) equipment, communication networks, buildings or facilities, personnel and the computer programs. To attempt to simplify matters, we shall confine our remarks to the design of the computer programs at the system level recognizing, however, that the programs must successfully interface with the other major system segments in the operational environment.

In a large-scale military system development project, the design of the various system segments may be the responsibility of different contractors. Let us assume for our discussion that one contractor is designing the entire system. Or, as is commonly the case in industry, the computer has been selected and the systems men are only concerned with developing the computer programs involving system inputs, data processing and outputs.

A major design effort which occurs at this point is the allocation of operational functions (accounts payable, inventory control, billing, payroll, planning, etc.) to equipment, human beings or computer programs. Here we encounter

such issues as whether a human being or a computer program should make a decision of a certain type. In the case of planning we may assign this task to a human operator; in the case of inventory control or billing we may assign it to the computer programs. System design brings us to the crucial area of interface between human operators on the one hand and the computer programs and equipment on the other.

A number of inadequacies in the training of the systems man becomes apparent at this stage. These include: (1) systems engineering for the total computer program system, which in a large-scale system may be composed of hundreds of separate programs and subroutines; (2) test planning against performance criteria in the system specifications; (3) personnel related features such as human engineering, task analysis, personnel requirements, training requirements, and job and organizational design, and (4) the writing of design specifications.

1. Systems Engineering

The systems analyst who has graduated from the rank of computer programmer is unlikely to have had adequate training in the wide variety of available systems equipment — not only computers but all manner of input and output devices, consoles, displays and communications. While this hiatus in his training background is understandable, it is more difficult to accept inadequate preparation for the systems engineering of the total computer program system by itself. The fact is that it is much simpler to talk and write about the virtues of a systems viewpoint, of designing total integrated systems, than it is actually to design and produce a system. If the system under consideration is composed of, say, one hundred different programs, one must inevitably fragment the total design task among groups of designers. These groups are usually assigned responsibility for different operational functions — in military systems such functions as air surveillance, tracking, weapons control, etc., and in business systems such functions as payroll, billing, production control, etc. The designers tend to become specialists over time in one or another of these various functions. The consequence of this fragmentation of tasks and specialization by functions is that comprehension of and control over the total computer program system tends to be dissipated, if not lost altogether. The ultimate dismal result too frequently is that the system after it is produced fails to meet the system specifications. A basic system specification such as computer operating (cycle) time may not be met; or reports may not be timely, or accurate enough, or relevant to a decision-making function, despite the fact that these requirements are stated in the system specifications. Appropriate training, it seems to the writer should make it possible for system people to anticipate such problems; to be alert to the consequences of job fragmentation and specialization; and to realize the need for some individual or group to conduct the systems engineering function throughout the course of system design.

2. Test Planning

One of the most difficult aspects of system design is to develop a test plan whereby both the contractor and the customer can be assured that the computer program system yet to be designed will, in fact, conform to the system specifications. For reasons which are not too clear to the writer, these are relatively neglected areas of training for computer programmers and systems analysts. By testing in this context the writer is not referring to the debugging which a program-

mer normally performs in the course of writing the simplest type of program but rather to the system level in which testing must be conducted to ensure that interdependent programs will, in fact, operate as a multiple set to perform a system function.

Where or when in his career is the typical systems man taught to design a test plan in which the customer will be able to trace the relationship between an operational function in the system specification and a particular computer program? Typically, as a result of the system design translation of system specifications, there is no one-to-one relationship between operational functions and computer programs. Yet the user has the right to know how each program satisfies some functional requirement. In the military area this question of "traceability" has become a critical one. It may be less of a problem in today's smaller-scale business systems. But as the trend toward "total integrated systems" in business continues, one suspects the problem of designing satisfactory test plans will also become increasingly important.

Task analysis is an essential preliminary step in the determination of personnel requirements, operating procedures, training requirements and job and organizational design. This is a highly specialized skill for which the systems man receives little or no formal training. This problem is addressed in the following section.

3. Personnel Requirements

In a previous section the typical organizational problems encountered in a system development effort have been touched upon and will not be discussed further here. People, of course, make up the organization. In large-scale military systems, the problem of people tends to be neglected by the programmers and systems analysts, partly because they have not been trained to consider people as components of the system. Problems related to people may be dealt with by other specialists known as "human engineers," "human factors" personnel, "training specialists" or by an outside consulting or training organization. These types of specialists handle such matters as task analyses, the design of operating procedures and the design of console switching arrangements and graphic displays. They prepare job descriptions and develop training requirements and training programs. They write training guides and user manuals. They may also conduct training operations.

This raises the question of who performs these skills and functions for smaller systems in industry, for example, when the funding of the development project does not permit hiring specialists or consultants of the types referred to above. Presumably in such cases, the systems analysts must be prepared to perform these skills and functions. On what basis does the computer programmer working on the design of a business system determine the allocation of functions to human beings or computer programs? Presumably such decisions would have to be based on an understanding of the capabilities and limitations of human beings. But it is not clear where the systems man obtains this sort of information in the course of his formal training.

4. Writing Design Specifications

In any system development effort there eventually comes a time when the systems staff must write a document which describes the system to be built. The design specifications state *how* the computer programs will perform the system

requirements. This is a traumatic point in the typical development process. The design specifications must be intelligible to two entirely different kinds of people: the customer or ultimate system users and the computer programmers. Thus it must be both written in English (if that is the user's language) so that the users will be able to understand how their needs for data processing will be met, and it must be written in logical and quantitative terms which can be translated into computer programs.

Certainly it is no surprise to anyone that computer programmers as a group are not particularly skillful writers. If they were, they might have chosen some other profession. And when they graduate up the career ladder to systems analyst and data processing manager, there is no noticeable improvement in their linguistic skills. Here we have a formidable gap in training. The writer is not only referring to the fact that the secondary schools and the colleges could do a better job in teaching Americans to write their own language at least as well as Europeans who have studied English appear to be able to do. We want to emphasize here that the importance of adequate and clear documentation describing the results of system development work is not stressed nearly enough in formal courses of training for systems people. Programmers seem to have the notion that as long as *they* understand how their programs operate that will be sufficient. Nothing could be further from the truth. The systems man who cannot document his work effectively at the system design level will buy trouble from both ends – from the customer who may not understand how the design will satisfy his operational requirements and from the programmer who will design the computer programs to suit himself, not the designer. Of course, in relatively small system development efforts, the system designer and the programmer may be the same person. This will alleviate part of the problem but not all of it. The customer will still have to be satisfied, and he will want to read about his system in a language *he* understands.

PRODUCTION PHASE

In this phase the computer programmer has the responsibility of translating the system design specifications into computer program specifications and for the logical design, flow charting, coding and testing of the computer programs themselves. Debugging of individual programs and testing of assemblies of interdependent programs constitute a major proportion of the work conducted by programmers during this phase. Despite this fact, the training of programmers focuses largely on program design, as such, rather than on the testing process. The consequences of this distorted emphasis is a tendency to neglect, in a large-scale development effort, the design and development of appropriate test tools, test procedures and test documentation.

In a large-scale system development project it is not uncommon for the designers and the programmers to be two different groups of specialists. In this situation, the test plan may be written by the designers (during the design phase) while the actual tests are conducted by the programmers (during the production phase). Unless there is very close coordination between the designers and the programmers, one consequence of this situation is that actual test outputs may fail to match the outputs called for in the test plan. The typical task fragmentation and specialization one finds in large contemporary "software" organizations may make such coordination difficult if not impossible. The writer suspects that if the training of systems men stressed the importance of the testing function more

than is currently done, this type of problem would occur less frequently and would be less serious. One might find, for example, that the designers had provided the programmers with appropriate test tools, such as simulation programs, early enough in the development schedule to facilitate program testing. In addition, such training would make designers more sensitive to the important role played by good and complete test documentation in successful computer program testing. As matters now stand, test documentation is regarded as a secondary and onerous chore, whereas program design is regarded as the primary and most rewarding task. But a computer program with errors in it is obviously of no use to anyone. At a minimum, test documentation should include a test plan, referred to earlier, a document describing the test procedures to be employed in conducting the test plan, and a document describing the test results with recommendations for changes. The latter two documents, of course, are prepared during the production phase. The total subject of test documentation should play a major role in computer programmer training.

Another major problem during the production phase is writing a complete and accurate description of the computer programs that have been produced and then maintaining that description as the programs continue to change throughout the production process. As we have noted, systems composed of computer programs are characterized by experimentation and evolution. It is commonplace, as noted earlier, for the customer to request new programs, modifications to programs, new operational functions or applications and even to introduce major equipment changes while the production phase is in progress. Thus, in addition to the problem of teaching programmers to be able to describe their programs in intelligible language, to which we have already referred, there is the additionable requirement to teach programmers the subject of "configuration management."

Configuration management is a term used in the Department of Defense to refer to the processes of identifying, accounting for and controlling contract end items, including computer programs. Nonmilitary systems have the same requirement for configuration management, although their designers may not use that expression. The processes of identifying, accounting for and controlling computer programs begins during the first system development phase and continues throughout the life of the particular system. It is during the production phase, however, that major difficulties are encountered in maintaining an accurate and complete system description. Despite the importance of configuration management for all types of systems, this is a subject that is not typically taught in formal courses for computer programmers.

INSTALLATION PHASE

The installation phase is that period of time between the production of the system elements and the beginning of operations using the new system. Whereas military systems may allow for a period of installation prior to the operational date, business systems are more likely to be installed in the midst of on-going operations.[9] The comments in this section are particularly relevant for the latter type of situation.

A typical installation phase for a large-scale system will include the following activities: (1) instruction of the users in the new system's characteristics; (2) a period of trial operation of the system by the users with the assistance of the

system's developers, and (3) a period of test and evaluation of the system's operations in the live environment against the requirements documented in the system specifications.

The outstanding feature of the installation phase is the high frequency of interaction between the developers and user personnel. In this respect, the phase is comparable to the initial requirements phase. Thus, skill in human relations, in addition to technical expertise, is a primary requirement for systems people.

A major source of potential tension between systems and user personnel at this point is the presence of errors in the computer programs. Typically the users do not expect to receive computer programs which contain errors. Unfortunately this is what they frequently get. Experienced programmers know that despite extensive testing during the production phase, the programs will not be error-free. Some errors will inevitably be found when all the programs are run in the operational environment. The most thorough testing will not anticipate all the outputs which are potentially possible when a large number of programs interact.

In addition, inexperienced user personnel are now also interacting with the computer programs, and they will contribute their share of errors. Additional errors may be introduced as a result of equipment malfunctions. Locating the true source of system errors may become a time-consuming and laborious process. Hard feelings may be commonplace as the users blame the computer programs for errors which could not be anticipated; as they blame the computer programmers for errors caused by inadequately trained users; and as they attribute system failures caused by equipment malfunctions to the computer programs.

The point here with respect to the training of systems people is that they not only need to be prepared for their day-to-day relationships with the users, but they must also be trained to be able to design and implement orientation and training programs for them. The need for such programs will, of course, vary with the level of user experience with computerized systems. Initial difficulties with new computerized systems have in the past been frequently due to inadequate orientation and training of user personnel. These difficulties might have been avoided, and may be avoided in the future, if the systems people are trained to recognize their responsibilities to prepare the users in advance for a trial period which will ensure a smooth transition from production to operations.

Since the new system will most likely alter relationships among individuals and groups in the user's organization, the installation of the system can be expected to arouse feelings of anxiety and insecurity. Outright resistance to the use of the new system is not uncommon. The orientation program should have as a major goal the relief of such feelings through an educational process which not only describes the new system and its operations but also clarifies new organizational procedures and interactions. Alterations of individual status as a result of computerized operations will not be taken lightly by the users, especially if they find themselves lower in the social pecking order than they were before the system was installed. Here again we touch upon the issue of sensitivity to organizational factors which should be a recognized and prominent part of the training of systems people.

With the beginning of operations in the live environment, the system "as built" can now be tested and evaluated against the performance criteria in the system specifications. Events have shown that large-scale computerized systems are, as we have noted, experimental and evolutionary in nature. If this is so, the

skills and knowledge the systems people bring to bear at this point in the system's development should include experimental design and statistics.[10] The system designers should be able to design experiments to prove, for example, that time-sharing is more effective and less costly than batch processing for a particular type of operation, or to demonstrate that a change to the executive routines may substantially reduce program operating time, etc. Indeed, the entire subject of system evolution under planned control by the system's developers and/or users deserves a prominent place in the curricula provided for systems people. This is not now the case.

CONCLUSIONS

1. In addition to the determination of training requirements for systems personnel by job categories, this paper suggests that another dimension of such requirements can be obtained by analysis of the attributes of typical system development phases.
2. One may conclude from the approach taken in this paper that the education and training of systems people should be a lifelong continuous process rather than a onetime thing. Phases of education and training need to be created which match in content and timing associated phases of system development. It should not be assumed that because a programmer has been successful during a production phase he will be equally successful during a system design or requirements analysis phase.
3. Personnel specialists ought to conduct research to determine the criteria which would lead to the selection of systems people suited to perform most effectively in each of the system development phases.

REFERENCES

1. M. V. Higginson, Managing With EDP: A Look at the State of the Art, American Management Association, Research Study 71, 1965, p. 27.

2. F. K. Bangs, *Curricular Implications of Automated Data Processing for Educational Institutions,* University of Colorado, Boulder, Colorado, September 1968, pp. 11-12.

3. Business Automation News Report, March 17, 1969, p. 2.

4. A sensitive version of this report has been written containing data referred to in this paper. Neither the original report nor the sensitive version will be cited here both because of the sensitive nature of the material and the fact that the documents have not yet been published.

5. P. E. Rosove, *Developing Computer-Based Information Systems,* New York: John Wiley & Sons, Inc., 1967, pp. 17-18.

6. M. V. Higginson, op. cit., p. 35.

7. D. Granick, *The European Executive,* Garden City, N.Y.: Doubleday & Co., Inc., 1962.

8. *Ibid.,* p. 239.

9. A military system similar in this respect to business systems is the National ADP Program for AMC Logistics Management (NAPALM); cf., *NAPALM Plan and Progress Report,* Washington, D.C.: U.S. Army Materiel Command, 30 September 1968.

10. For a detailed review of this subject see H. Sackman, *Computers, System Science and Evolving Society,* New York: John Wiley & Sons, Inc., 1967.