to the IBM 1401. Similarly, the standards for the 650 programmer, who corrected his errors through examination of the console, could not be related directly to the 7070 and 1401 programmers who corrected their programs in a much more sophisticated manner by examining memory through print-outs.

At the same time, operating software has increased in complexity to a point where the interface with the programmer is so different that a totally separate level of programming competence is required to understand it. This rapid technological change has impeded us in developing the necessary management capability. We have, in effect, been trying to hit a moving target whose velocity is continually changing.

It is easy to summarize the problem. We do not really know how to select programmers, and we tend to select those with some undesirable characteristics. Having selected this group of programmers, we usually provide them with ineffective, limited training, and little in the way of effective disciplines or methodology. Most often, each programmer is then capable of deciding on his own methods of communication and of maintaining absolute product control, all the while increasing his value in the industry and in the marketplace faster than his salary can be adjusted internally. Typically, he works for a manager who is ineffective because he has been given neither proper management training nor basic tools and disciplines with which to work; whose functions have not been defined, and whose process of communication with the systems analyst or user is generally confused. Finally, all this takes place within a technology which changes so rapidly that it is almost impossible to get a fix on the functions and the method by which the work is supposed to take place, before it changes.

Conclusion

Our objective must be to develop effective economic standards which allow us to measure and control the programming effort. This will obviously be a difficult task, which can ultimately be accomplished only if it is done for all levels and types of programming, for all types of installations, for all types of machines, and for all parts of the data processing community. If it is not done in concert, but merely in selected installations, programmers will simply avoid those installations and obtain jobs somewhere else. Concerted action on the part of the data processing industry is mandatory if efficient computer use in the 70's is to be achieved.

The solutions therefore lie in joint effort—toward definition of the functions, toward better selection practices, toward better and more realistic training programs, and, of course, toward the establishment of standards. The types of management difficulties which exist today cannot be allowed to continue into the future. If we can muster the resources, and apply them intelligently, we have an excellent chance of establishing economic control over the computer programming function. Barring this, the outlook is bleak indeed.

REFERENCES

1 *Historical development of the computer market*
Moody's Computer Industry Survey Winter 1967-68

Trend of the industry
Moody's Computer Industry Survey Fall 1967

2 G BYLINSKY
Help wanted: 50,000 programmers
Fortune Magazine March 1967

3 *The state of the information processing industry*
AFIPS Report May 1966

4 D H BRANDON
Jobs and careers in data processing
Computers and Automation September 1966

5 J L HUGHES W J McNAMARA
Programmer aptitude test—revised edition
International Business Machines Corporation

6 R A DICKMANN J LOCKWOOD
1966 survey of test use in computer personnel selection
Computer Personnel Research Group

CARL H. REYNOLDS
Computer Usage Development Corporation
Mt. Kisco, New York

The common complaint among people who must plan for and manage the development of computer program systems is that the products are almost always finished over budget and late, and they hardly ever do what they were intended to do. This paper suggests that the aforementioned state of affairs is not only undesirable but unnecessary; that there are certain relatively simple precepts which—if managers could become convinced of them and adhere to them—can be used to avoid many of our present difficulties.

As the computer programming field continues to grow and mature, it is becoming more and more apparent that:

- there is no **fundamental** difference between the process of managing the development of computer programming systems and the process of managing comparable developments in any other technology; and,
- moreover, that the managers who complain most vociferously about the alleged uniqueness of computer programming are usually the ones who spend the least time trying to understand the particular technical environment (or the technical people who do the job).

I contend that the management of computer programming is fundamentally similar to the management of any other difficult technology. First of all, it requires an understanding of the technical methods being used, and secondly, it requires differentiations depending on the "maturity" of the technology that is involved. I contend that these differentiations have been made effectively in such fields as engineering for years with notable successes (and notable failures). The same will be true in computer programming. In short, the problem is not entirely within programming, it is that programming is really many things and only rarely do people differentiate between the various levels of the technology and adjust their management methods and their expectations and their commitments accordingly.

Many people have said that the ordinary techniques of management cannot apply to computer programming. I suggest that the reason they appear not to is due to the fact that on the surface there does not seem to be anything to apply them to. In almost all other activities that merit any significant management attention there are things to count, things to cost, and times to be measured. In short, the status of most processes which management worries about can be determined by physical measurements and counts—whether it be dollars or numbers, buttons punched out of the machine per day, item counts in the inventory, or what have you. These numbers are harder to come by in computer programming and, therefore, the assumption is made that one cannot have numbers and, therefore, one cannot manage in traditional ways. I would say that while it is difficult, it is far from being impossible.

Let me give just a few examples of the kinds of physical milestones that can be established in programming and how they can and should be used to manage by normal, old-fashioned techniques.

- Establishing a plan of action to achieve a goal within a certain time period, and investment.
- Measuring, during the course of the project, the performance against that goal.
- Making evaluative decisions each time the plan and the actual performance mismatch.
- Taking corrective actions iteratively until the job is done.

The foregoing represents a procedure for achieving management control whether one is building a building or writing a compiler. Consequently, the fundamental management task, as I see it, is to translate the programming production process, at least partially, into some physical events or milestones.

- First, one has to assure that a relatively concise representation of the work is prepared. In broad terms, this can start with a one-page system description, which can then be expanded into, say, several simple subsystem descriptions. At some point, a document, which one can call External Specifications, needs to be produced. This can be followed by detailed programming and subroutine specifications, and the actual coding. These pieces of paper are the physical events which need to be managed in the production of programming systems, and they need to be managed in binary ways—that is, they either are completed by a given time or within a certain cost or they are not. In developing a plan, management must concentrate on physical events which have actually transpired and ignore the percentage completions of individual documents and programs. The "transfer" from one step in the process to another should be accomplished at the time that the particular part of the job is completed.
- It is vital in the management of programming that the second step—which is to measure progress against the plan—be made before the third step, which is the evaluation of the reason for departures from the plan. Time after time one gets a status report which shows that something is late, or behind, or over budget. Before getting psychologically prepared to deal with that problem one must listen to the harrowing, heart rendering—although often true—story of why things are the way they are. Consequently, one often slides through steps three and four before adequately judging the status.

By making use of these well-established techniques—which really amounts to "doing their homework,"—managers can greatly increase the effectiveness of their control over the computer programming process. At the same time, there cannot be any doubt that, while there are fundamental similarities between the management of computer programming and the management of other, more traditional processes the management of computer programs is, generally speaking, unusually difficult and poses a considerable challenge. This is the case for several reasons.

- First of all, in less than 10 years, programming has gone from an interesting, and sometimes useful, intellectual activity to a point where the defense and much of the economy of the Country literally

hangs on the commitment made in programming development activities. The responsibility that falls upon the shoulders of programming managers is out of all proportion to the maturity of the field and usually to the maturity of the people who bear the responsibilities. (Of course, in the long-run, it is recognized as being part of the management job to develop that maturity.)

- Secondly, it is not that easy to perceive the technological stages of programming, so that one cannot expect to manage computer programming without spending a considerable amount of time trying to learn about it. No matter how good a "manager" one is, the more one knows about programming, the easier it will be to apply management skills to the task.

The implications of the idea that programming has a technology were presented to me by Mr. George H. Mealy, formerly of Bell Labs and IBM, and now a consultant in Boston. Since I believe it is a key idea, I want to be sure to give him credit for it.

Webster says that technology is a technical method for achieving a practical purpose. The programming techniques used to implement computer-based systems satisfy this definition. Any particular technical method, whether it be the use of pre-stressed concrete in buildings, the use of transistors in electronics, or the use of SYNTAX-directed methods in compiler writing, goes through stages of relative practicality. For instance:

- In the **research** stage, work on a technical method is usually considered to have merit if a purpose can be achieved at all.
- The technical method, if it is fundamentally good, then progresses to a second stage which we might call **developmental**. In this case, practicality implies not only that the purpose can be achieved but that, at least in some instances, the purpose is achieved economically. For example, a technical method might not generally be economical but in relatively limited and special situations it may be the only way to achieve the purpose under some other constraint. An example of this, to me, is the fact that the development of SABRE could be justified long before general real-time, on-line inventory control was generally economical.
- Finaily, a technology or technical method can become **operational**, by which time it will have proven that it is capable of achieving a practical, economic purpose in a wide range of situations.

To summarize, computer programming has a technology: this technology consists of the technical methods used by programmers to achieve the practical purpose of making computing machines do useful things. These technical methods go through stages of development and application.

The key to my fundamental premise is that **the management of computer programming is more closely related to the management of technologies in their various stages of development than it is to the particular technical methods that are involved.**

So, I contend that the first thing that must be done to understand computer programming management is to recognize that there are different technological levels of programming and that the demands made upon management in terms of these different levels are substantially different. If one does not recognize this fact and does not take the trouble to try to figure out just what is being managed, then one is likely to fail.

Now, what does it take to manage computer programming in each of the three areas of technological development?

- In the "research" and the "developmental" stages a manager's principal concern ought to be about whether he—either by himself or through his advisors—has access to the detailed technical knowledge and experience necessary to make the decisions which must be made. Alternate paths must be selected; objectives must be evaluated. These are the same problems which confront managers of hardware research programs. Experience is needed which is different than that required for making decisions with respect to well-established operational systems. Moreover, the funds involved at this stage ought to be considered risk funds and managed accordingly. Also, one ought to keep the salesmen away. The best computer program in the world—and the easiest one to sell—is the one that has not been written yet. It is even better if nobody knows whether it can be written at all since many wonderful things then become conceptually possible.
- In the "operational" stage, a manager's attention is focused more on the detailed matching of the technology with the precise needs of the users than with the technologies themselves. This requires that the manager understand technology and push it to meet legitimate requirements. In this stage, the sales approach is not only valid but necessary to ensure the sales approach is not only valid but necessary to ensure responsiveness to the users' requirements.

Thus, I see that the first thing that the management of computer programming requires—which is "old hat" in more mature industries—is an understanding of what can be done with the current technology, and how to make commitments in terms of dollars commensurate with the risks involved in achieving the necessary system results. In these terms, computer programming can be just another management problem—except that

to make it so, one has to spend a lot of time and effort.

- First, one must understand computer programming well enough to know what is possible, what is probable, and what is impossible or unlikely; i.e., one must know the technology with which one is dealing.
- Second, one must make commitments based on the technology used, not on the needs of the world—and not on the unreasonable hopes of starry-eyed experts. Similarly, one must be sure to get enough, and as much as one can, out of the technology and not let laziness or incompetence get in the way.
- Third, one must insist upon schedules based on physical events, and on numerical descriptions of the product that are being produced, to the greatest extent that ingenuity will permit.
- Fourth, one must objectively assess the status of the project against a well-developed plan and decide for oneself the status of that project before attempting to understand what is wrong and what ought to be done about it.
- Finally, of course, one must do something about the trouble one finds.

Thus, given these prerequisites, I conclude that computer programming can in many respects be managed just like any other process.

CARL W. CLEWLOW
United States Department of Defense
Washington, D. C.

The rapid strides that are being made in the physical sciences and the relatively slower advancement in the administrative arts and sciences have worked together to create an ever-widening management gap. Perhaps nowhere is this gap more apparent than in the field of computer management, where the physical development of the computer has outstripped management's ability to come close to achieving an optimum or even a moderately effective utilization. My purpose in what follows is to explore some of the circumstances that have caused and maintained this gap, especially with regard to computer programming—but, more important, to identify those efforts and techniques which have given promise of its narrowing.

While my remarks will be focused primarily on the Federal Government, this is not to say that the Federal Government has a corner on all of the problems in this area—or all the immediate solutions.

Some Dimensions of the Computer Programming Management Problem at the Federal Level

Before proceeding further, it might be well to identify some of the major problem parameters and organizational characteristics that must be considered in managing computers and computer programming in the Federal Government.

Organization

Despite many views to the contrary, the Federal Government cannot be viewed as a monolithic structure wherein all problems, programs, and processes respond to the same or even similar demands or impacts. Indeed, there are strong cross-currents and countervailing forces within the structure of the Government which reflect differing responses to the same stimulae—such as budgetary needs, appropriations, program objectives, and many others.

Computer Installations

There are nearly 3,800 computer installations in the Federal Government, ranging from very small to very large, that are engaged in administrative and research activities. There are another 2,500 computers in the hands of contractors in the private sector, paid for by the Federal Government, working in support of contracts for producing end-items, assemblies, and components. Neither of these figures includes the substantial number of computers devoted to classified activities.

Number of Computer-Associated Personnel

ADP man-years in the Federal Government have grown from a mere handful in 1950 to more than 90,000 today. Indicative of this growth, and the large concentration in the Department of Defense, are the following data:

TABLE 1. ADP Man-Years in the Federal Government

Year	ADP Man-Years in Federal Government	ADP Man-Years in DOD	% of Federal Government Man-Years in DOD
1964	73,310	51,873	70.6%
1965	76,158	55,170	72.4%
1966	88,953	59,778	67.2%
1967	90,589	63,799	70.4%