FORMAL AND ON-THE-JOB TRAINING FOR APPLICATIONS SYSTEMS PROGRAMMERS

Mr. J. D. Aron

Federal Systems Center,
I.B.M. Corporation,
18100 Frederick Pike,
Gaithersburg,
Maryland 20760, U.S.A.

Abstract:

After a discussion of the current methods of producing large software systems, more appropriate forms of organizing and training programming personnel are suggested.

Rapporteurs:

Mr. J. S. Clowes
Mr. I. Mitrani

The growing requirement for very big programs gives rise to problems connected with the efficient organization and management of the large teams of programmers needed to produce them. Techniques which have been successful in the past, when programs and teams were smaller, have proved inadequate for the largest present-day projects. In his lecture Mr. Aron analysed the reasons for the inadequacy of current practice and put forward suggestions for the more appropriate organization and training of programming personnel.

In order to provide a context for his later remarks Mr. Aron commenced by describing the nature of the business and the type of person employed at his institution.

The IBM Corporation's Federal Systems Center is, in effect, a large software house whose business consists in writing applications programs for its customers, chiefly the Federal Government. The programs are predominantly non-scientific and non-experimental, that is, they are required to perform well-defined tasks and will be run on a routine basis. An important factor is that most of the users have limited technical knowledge of computing and the programs must be designed so that they can be easily and reliably used by such people.

The size of programs produced has varied from 10,000 to 6,000,000 steps (i.e. macroassembler or high-level language statements). Programs of 10,000 steps would be classified as 'small', those with 30,000 to 500,000 steps as 'medium' and those with over 500,000 steps as 'large'. This paper is concerned with programs in the 'large' category of which about 12 have been attempted.

Most of the programmers are graduates, the majority of them in their first job. Very few are Computing Science specialists, the national output of such being comparatively small. College graduates are selected primarily because of the belief that they are more likely to succeed in intellectual work than non-graduates. On the whole, they do not have, and do not acquire, a professional attitude towards computing. Programming is regarded just as a job and most are unwilling to deepen their understanding of the subject through spare-time study. The 10-20% who do study tend to advance more rapidly. (It should be noted that even 10-20% of a population of several thousand is a very respectable cadre of software engineers.)

In general, one may distinguish between training, which consists in teaching facts, and education, which consists in explaining concepts. Since the cost of training and education increases the price of the product, instruction in industry tends to concentrate on training which is more immediately useful. Even so, the time which may be devoted to training is very limited and employees at the Federal Systems Center receive only the equivalent of one week's training per annum on average. The employees are expected to supplement the training with graduate level education courses and technical society activities on their own initiative.

Training in some particular subject is, obviously, best provided immediately before an individual is assigned to a project involving the subject. In practice this is usually impossible because courses are scheduled to take place at fixed times. Conversely, it is often impossible to release an individual from his current project at the time when a course is running. The result is that the only training a programmer can be certain of is his initial training consisting of nine weeks programming plus on-the-job instruction. It is through on-the-job training that most advanced techniques are taught.

In I.B.M. team managers are not only responsible for the success of their projects but also for the progress and growth of the team members. It is essentially a leadership job and team managers are team oriented. A large portion of the present managers were trained as engineers and have been at the center for up to ten years. Because of the pressure of day to day work they have had little time to study the development of computing techniques. Aware of the overriding necessity to finish projects on time and within budget they tend to be conservative in their choice of methods and to prefer modifiable, easily understood and essentially simple programs. Trainees on the other hand are self-oriented, interested in experiment and in developing their own ideas. They prefer to seek elegant solutions to problems. This divergence of outlook tends to produce conflict and one of the aims of training must be to resolve this conflict. In order to do this it is necessary to train the programmers to work in a team and the managers to be more receptive to new techniques. At the same time, the training program must be designed to accommodate the existing manager-employee relationships.

Mr. Aron designated three stages in the historical development of commercial computing which have shaped the content of training programs.

During the first stage the computers, and consequently the problems, were small enough for one programmer to cope with. The man responsible for the application could grasp and understand the problem and he knew all about the tools available to him — such as assemblers, library of subroutines and Basic Input-Output Control System (see Figure 1). The programmer communicated directly with the computer and was able to take advantage of its peculiarities.

With the advent of the second generation computers larger and more complicated problems could be tackled. These were handled by a small team of programmers and although the whole team would contribute to the formulation of the problem, the actual programming would be divided into independent tasks. More tools were made available to facilitate the process of programming (see Figure 2); however, the team could equalize the training load by assigning each member a limited number of topics to specialize in. The proliferation of system programs and the increasing speed of computers necessitated the development of Operating Systems which, in this stage, relieved the programmer of repetitive chores.

The third stage covers the late 1960s and the present days. In the big and fast third generation computers the system program began to assume a more and more dominant role. It became so big and complex that the individual programmer could no longer know everything about it (Figure 3). Hardware developments (removable disks, communication attachments, manual input devices, etc.) and new tools and techniques made possible the solution of very big problems and very big teams of programmers were needed to do it. (Figure 4). The size of the projects implied that individual programmers could not understand the whole problem. They would make technical decisions which were logical at the programmer's level but illogical and perhaps catastrophic at the project level. The increase in the number of programmers engaged on the project led to a faster increase in the number of non-programming personnel with exponentially more complex interactions within the project.

At present, the number and variety of tools and techniques that programmers use are such that they present a considerable intellectual burden. No single programmer can effectively master them. There is a growing need for simplification of the programming process so that it again becomes feasible to train each man to be an effective programmer.

As one way of accomplishing this, managers of big teams concentrated mainly on being able to control the development of the project. This, in a

standard pyramidal organization involves the sub-division of the problem into independent units small enough for an individual to handle. These units are carefully isolated from the rest of the system and it is the system management team's responsibility to reassemble them (Figure 5). This method is called 'The System Management Approach'. It superimposes the system management umbrella on the organization without relieving the programmers of the need to know all the tools and techniques.

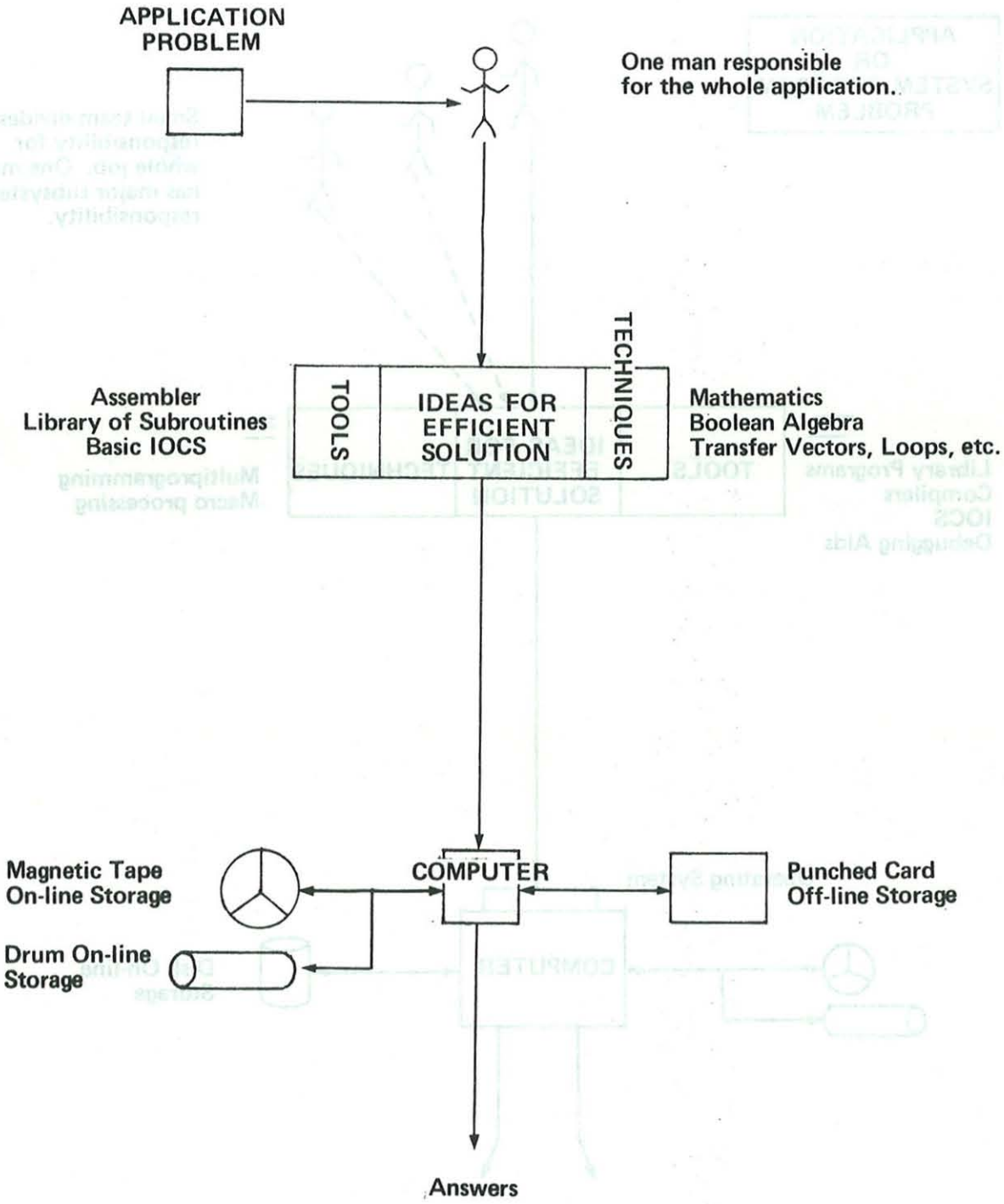
The system management approach benefits from the greater capability of new employees who have had computer education in school before joining the project. In addition, new development support tools, a blend of technical and managerial procedures, have been introduced to reduce the workload on individual programmers. Some of the tools are shown on Figure 6, where the abbreviations are as follows:

1. TSS - Time Sharing System - used mainly for interactive code editing and debugging and for fast computer turnaround.
2. JCS - Job Control System - used to collect, store, retrieve and link programs as they are released for testing and, eventually, operational use.
3. TSO - Time Shared Operations - used to merge the features of TSS and JCS to enable programmers to work on line from initial program construction through final test.
4. APADS - Automatic Program Analysis and Documentation - used to force the programmer to explain and comment on the program being developed.
5. Modelling and simulation of a system - used to make accurate predictions about system performance and to measure actual results.
6. PPL - Program Production Library - used for storing design information about the system as well as assisting in the collection, debugging, integration, and test of programs in process.

Assuming properly educated programmers and appropriate tools, an alternative approach has been proposed by Dr. Harlan D. Mills - 'The Chief Programmer Approach'. The chief programmer is a highly qualified individual able to take responsibility for defining, programming, testing and delivering a large system. Available to him are expert specialists who know all there is to know about a narrow field and who are able to answer questions and solve specific problems in this field. A team organized on these lines (Figure 7),

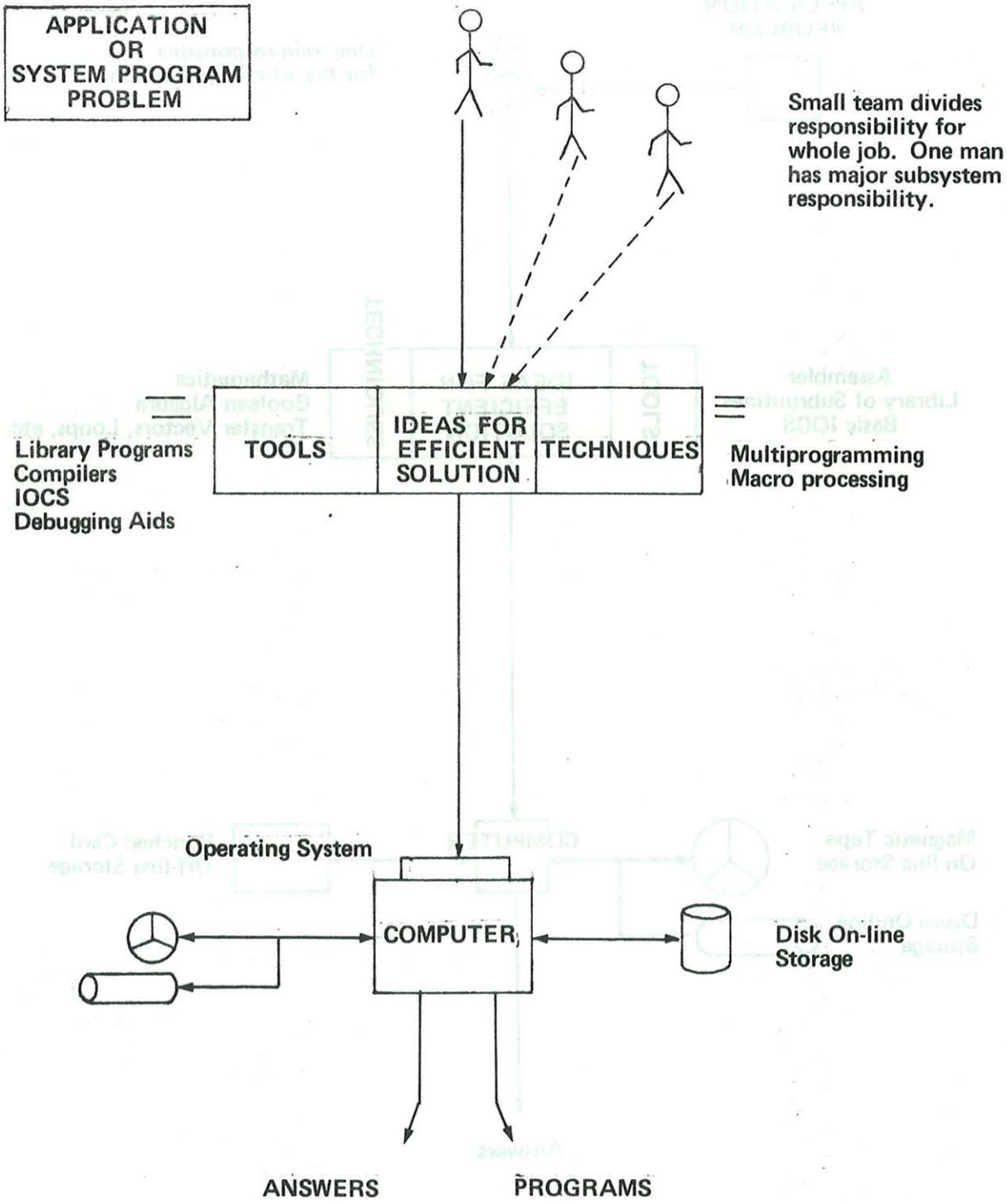
does not normally exceed ten people and will usually expect to accomplish the same results as a standard organization two to five times as large. Of course, some systems are too big for a chief programmer team to implement and they will still have to be handled with the system management approach.

In its future development, programming will consist of tasks, each of which is suited to a man who is best qualified to do it. The system management approach - which fits the problem to the man - can be employed in existing environments with proper training. The chief programmer approach - which fits the man to the problem - probably requires an education base in addition to job oriented training.



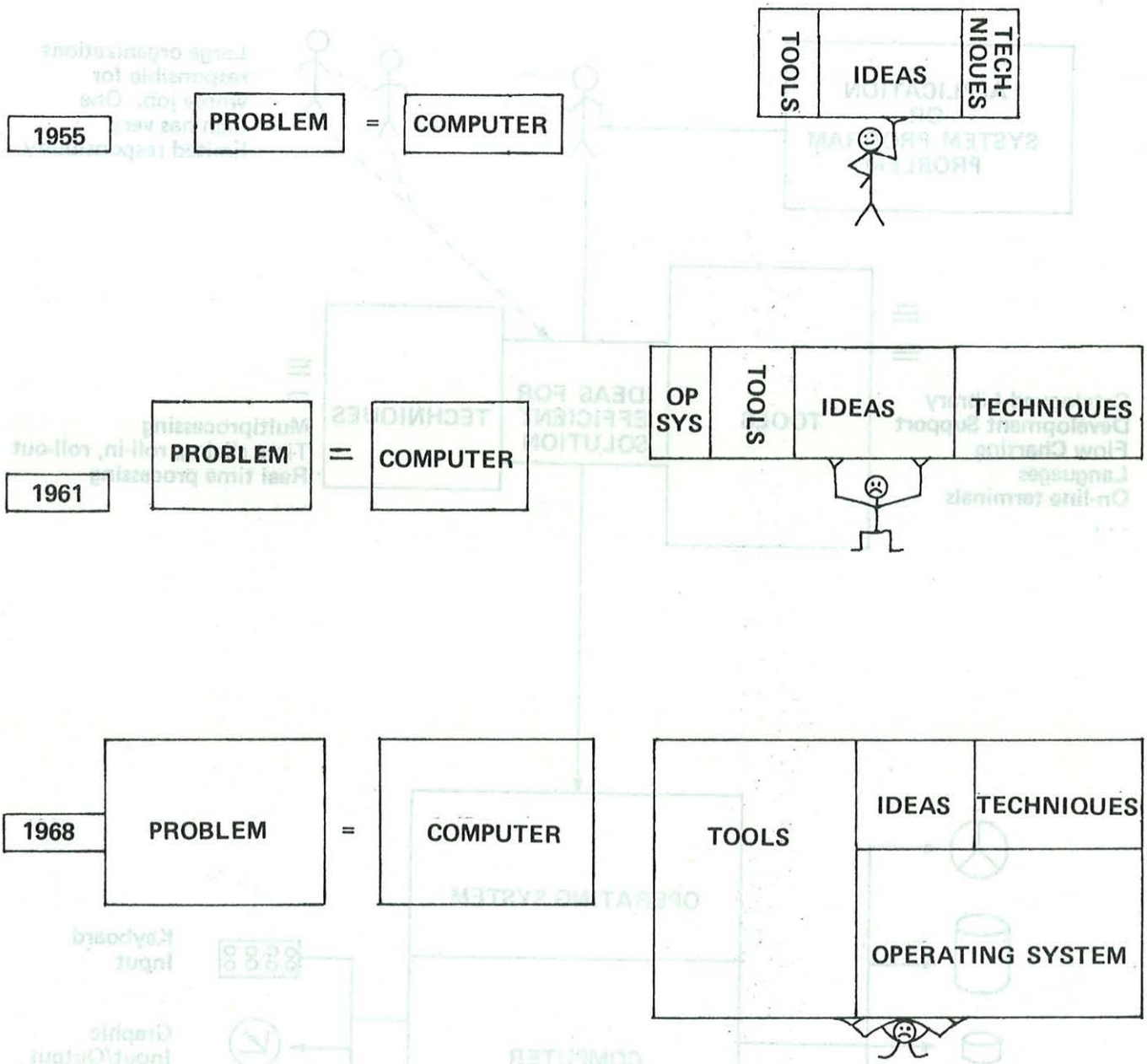
PROGRAMMING — FIRST STAGE

Figure 1



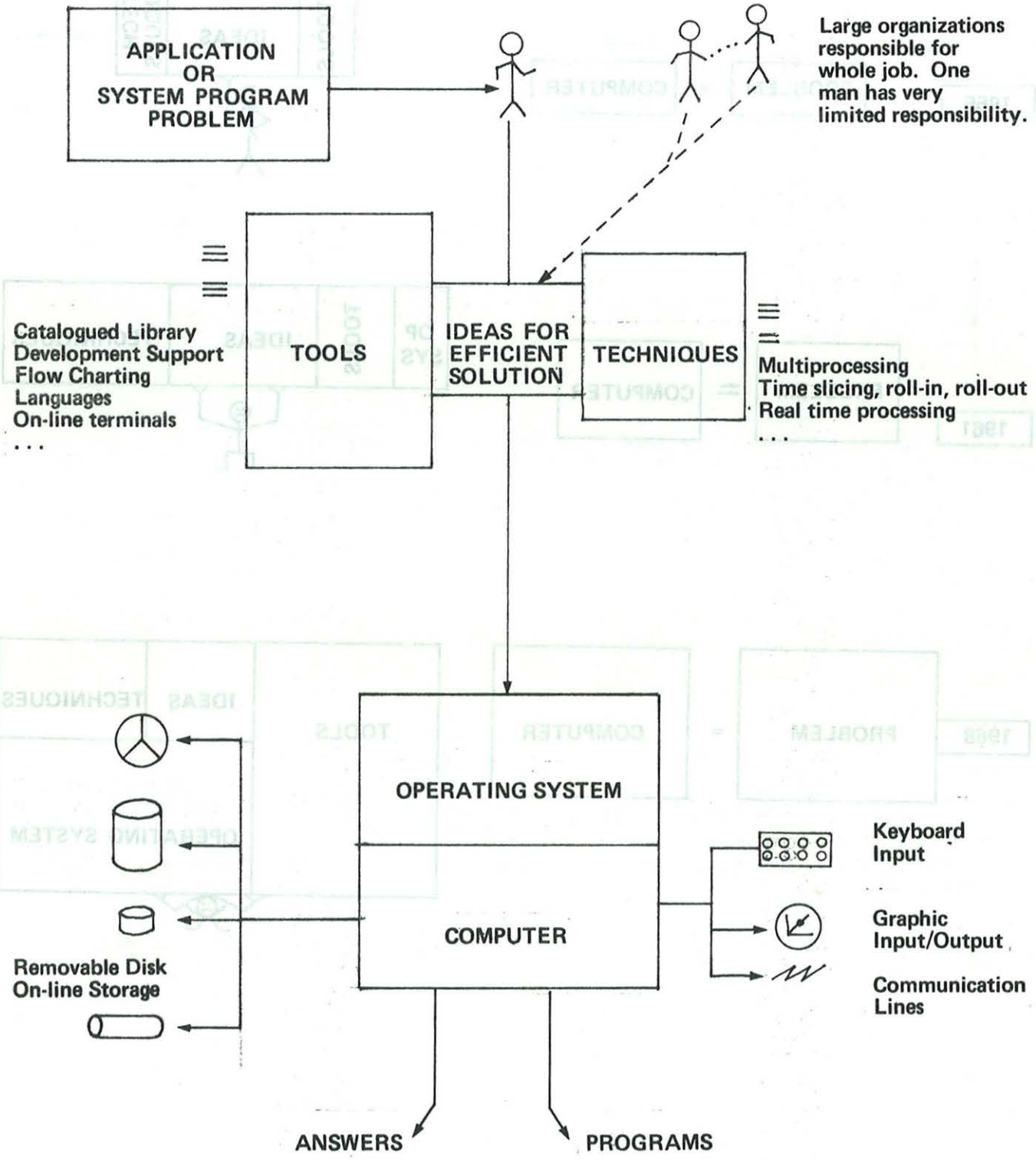
PROGRAMMING – SECOND STAGE

Figure 2



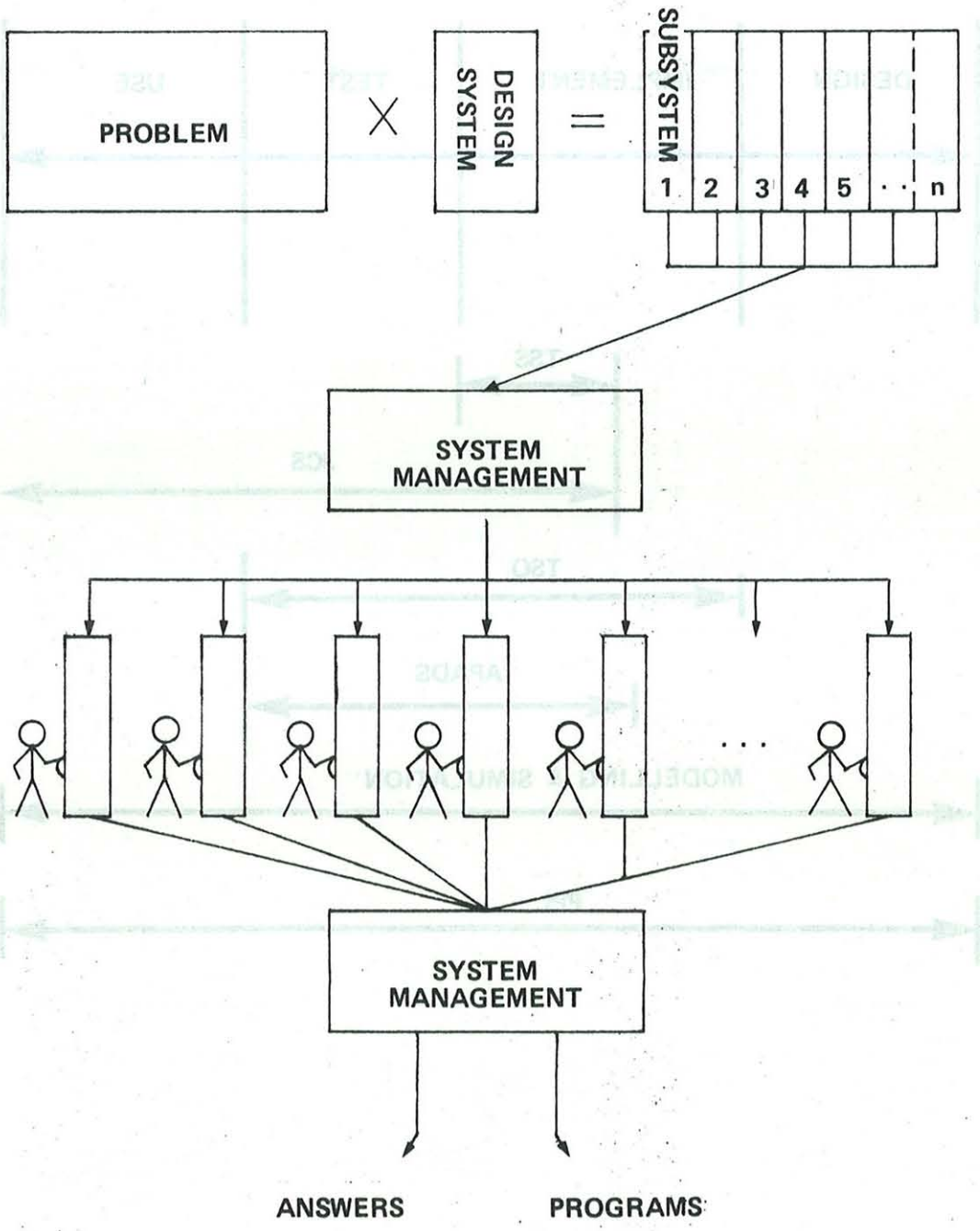
CHANGING NATURE OF PROGRAMMING

Figure 3



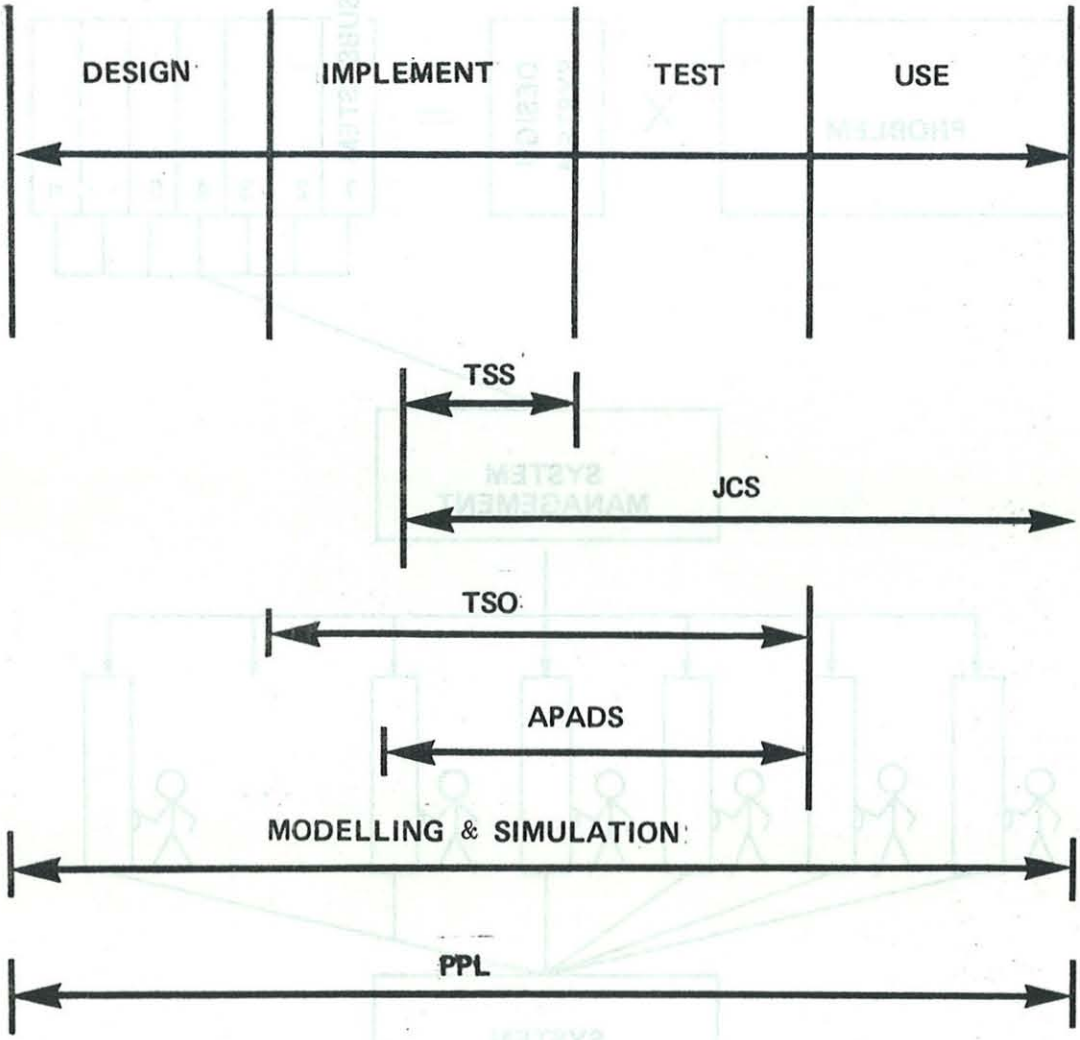
PROGRAMMING – THIRD STAGE

Figure 4



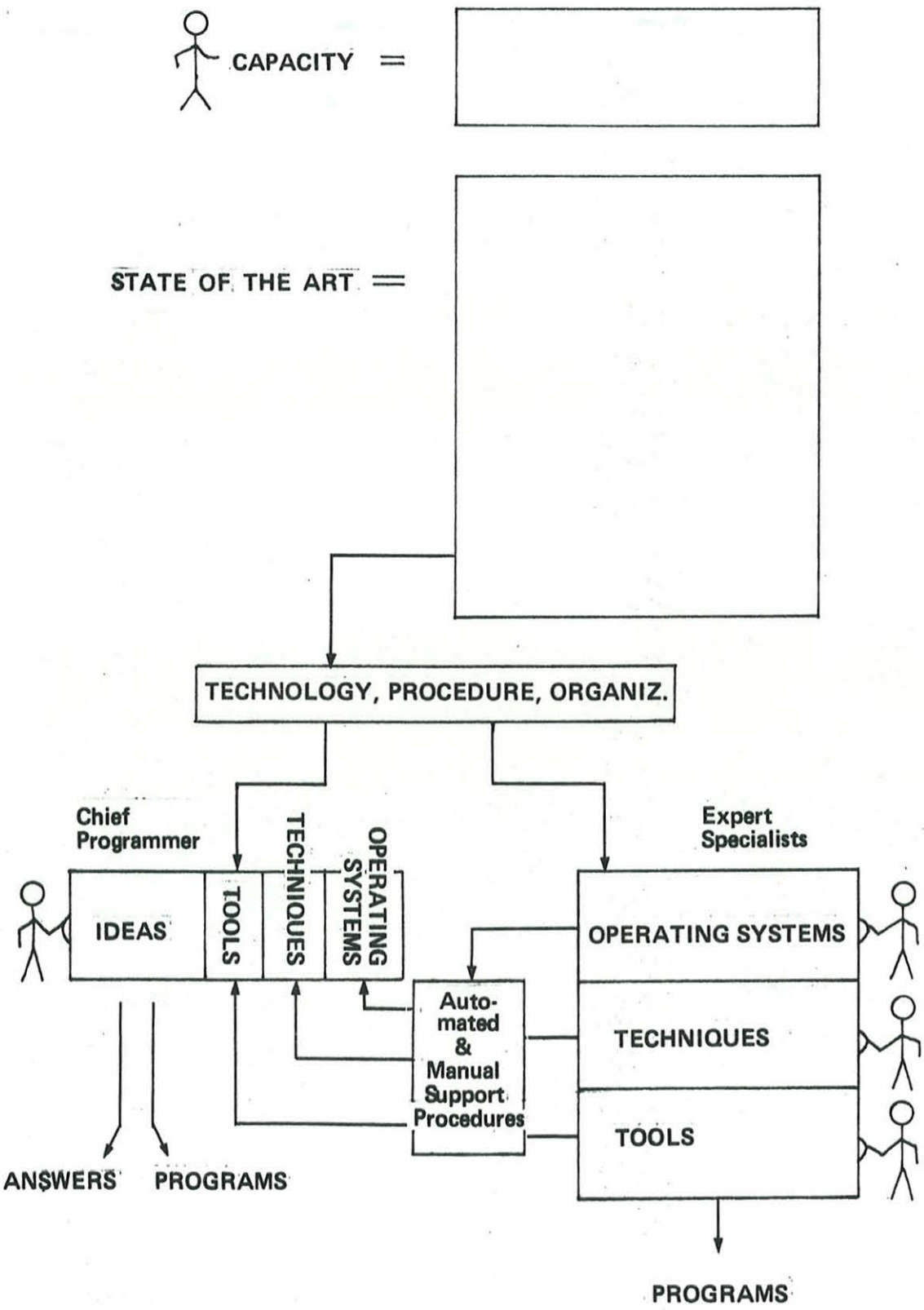
FITTING THE PROBLEM TO THE MAN

Figure 5



DEVELOPMENT SUPPORT SYSTEMS

Figure 6



FITTING THE MAN TO THE PROBLEM

Figure 7