N73-19214

DOCUMENTATION: MOTIVATION AND TRAINING OR AUTOMATION

Melba L. Mouton
NASA Goddard Space Flight Center

One of the eternally discussed problems in almost any technical or administrative setting is the problem of communications. The subject of this symposium indicates the concern about that problem in programming activities. In fact, it indicates not only concern but that many people are involved in trying to do something about it. Because of this deep concern and because of a tendency of many people in and out of electronic data processing (EDP) management to feel that anything and everything can be fully automated, it would be a good idea to consider what can be done to relieve the roadblocks or mental blocks in those areas where automation is not taking care of the basic documentation problem.

In the development of any sizable computational project, it is common to involve all the organizational elements that are related to the project design and use. These organizational, as well as functional, entities usually consist of people called managers, analysts, data analysts, and programmers. For the moment, consider these as the implementers of the project.

Table 1 indicates that for the first version of such a project, considerable attention is given to planning who will be responsible for developing the various manuals that make up

Table 1.–Original Project Documentation

Type of manual	Implementer	Users	Intended audience
Requirements	Analysts Programmers Managers Data analysts	Analysts Programmers Managers Data analysts	Managers
Analysis	Analysts	Analysts Programmers Managers	Programmers
Program specifications	Analysts Programmers	Analysts Programmers	Programmers
User's	Programmers	Data analysts Programmers Analysts	Data analysts
Programmer's	Programmers	Programmers	Programmers

the system documentation. On the left, the various types of manuals, requirements, analysis, program specifications, users, and programmers, are listed. In the next columns, the implementers, the users, and the persons for whom the document was written are listed.

The implementers for the requirements manual should involve all four disciplines; the analysts, programmers, managers, and data analysts are therefore listed. The users are normally a similar group of people; however, it is good practice to consider that it is being written for the manager.

The analysis manual is written by the analyst for the use of programmers, analysts, and managers. It is necessary for, and thus documented for, the programmers. Because of this need, some persons consider it as the first part of the next document, the program specifications manual. In any case, this specifications manual, too, is documented for the programmer and used by both programmers and analysts, the two groups that usually work together in developing it.

Next is the manual that is most often developed and has the least problem in implementer motivation and training, the user's manual. This is usually prepared by programmers for data analysts, but it is quite useful for the analysts and programmers as well.

Last, but not least, is the programmer's manual, written by programmers for programmers and, consequently, written with their knowledge or understanding in mind. This is the only manual for which the author, or the implementer, is the same as the principal audience and user.

Since the persons who need this manual the most are also the persons who are under the greatest pressure to get the program code written and debugged, it is not uncommon that this is the most neglected aspect of system documentation. Until managers, analysts, and programmers begin to make programming equivalent to planning, documenting, coding, documenting, debugging, and documenting, the checked-out, or almost checked-out, code will continue to be considered the end product of a programming effort. However, the goal of programmers' efforts must be both the checked-out code, the communication necessary for the machine, and the checked-out programmer's manual, the communication necessary for people, especially programmers, to continue to develop and maintain the system.

Documentation as a part of planning a program may have a relatively higher initial cost, but if the planning and associated documentation is done well, problems that could cost 10 times as much to solve later may never arise. In addition, for problems that do arise, solutions can be found and implemented more easily. It is difficult to determine who is most responsible for not allowing the time to do adequate documentation of the program, but it is clear that if programmers consistently give time and manpower estimates that do not include documentation, they will not be able to create a document that can give the recognition and reward that should accompany any good programming effort. However, the fault lies equally with those analysts or managers who accept a low bid in time and/or money for an undocumented or a half-documented system rather than a reasonable bid in time and money that allows for an up-to-date, good, clear description of the program. The time not allowed now will be allowed later, not for documentation, but for difficulty in program change and difficulty in the analyst user finding out what is in the system, as the continually changing but undocumented program persists.

For either or both of these reasons, the phase in which everyone is involved in this vital new project changes to what is commonly referred to as the programming maintenance

Table 2.—Documentation for Project Maintenance

Type of manual	Implementers	Users	Intended audience
Requirements Analysis Program specifications			
User's	Programmers	Data analysts Programmers Analysts	Data analysts
Programmer's			

Table 3.—A Comparison of Preliminary and Final Documentation

Steps in documentation	Preliminary documentation	Final documentation	Automation
Description of the problem	X	X	
Method of solution	X	X	
User instructions	X	X	
Flowchart	X	X	?
Subroutines used	X	X	
Program listing		X	X
Test documentation	X	X	

phase. Table 2 indicates what happens to the manuals that initially may have been adequately done. It only takes a "few" minutes to put in a "simple" change and check it out. Thus the cycle begins of quick change of code, but there is no allowance on the part of managers, analysts, or programmers to keep up what might have been reasonably good documentation. Thus begins the refrain, "it is documented, but it is not up to date." Not allowing the time needed to code and keep supporting documents updated comes at the point in the life of a system when programmers are the only ones seemingly responsible for updating all the relevant documentation, not just the programmer's manual. Thus, standards, guidelines, and, possibly, automation that will really have an impact on the entire problem of keeping computer program documents up to date must be considered.

The biggest problem, however, seems to be in the programmer's manual, and table 3 indicates what could and should be done as preliminary documentation. The only thing that cannot be done then is the listing. As soon as programmers are motivated and trained to plan (including the adoption of useful automatic procedures) and document prior to

coding, they will see how well it pays off, especially once it is realized that the manual is written by the time the listing is available. With the source code and the basic documentation available, more automatic procedures may be used for ease in updating and verifying a program and its documentation.

Because a good, descriptive planning flowchart is very important to the development of a program, delaying the generation of flowcharts until after the code is completed is detrimental to programming efforts. At this point, it should be emphasized that current automatic flowchart techniques should use the descriptions from a well-planned flowchart. The emphasis should be on how to produce planning block diagrams, flowcharts, and other documentation that can be updated automatically. A flowchart automatically determined from the source code without such preplanning is almost useless. This is best stated in the following excerpt from reference 1:

> Flowchart—this must be a structural flowchart of the sequential logic and decision points included in the program. Machine-produced flowcharts of the exact programming techniques cannot be used to satisfy this requirement as they merely amount to a listing of the programs and do not briefly and concisely reflect the inherent logical flow of decisions.

This refers, it seems, to "source-code-only" flowcharts.

Figure 1 is a simplified version of some information given in reference 2. It summarizes the importance and attention that must be given to documenting during the whole lifetime of a program or system of programs for good system development and maintenance. Figure 1 simply indicates that documenting, like management, must be considered an integral part of every phase of system development and use.

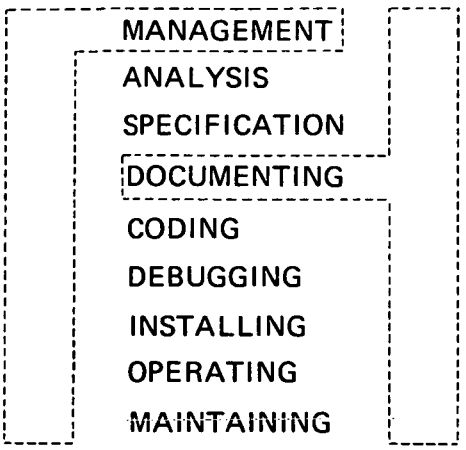

Figure 1.—The relative importance of documenting and management.

REFERENCES

1. Anon.: *Documentation and Program Standards Handbook.* COSMIC, Univ. of Georgia, Nov. 1968.
2. Control Data Corp.: *Seminar in Documentation,* Sept. 1968, p. 13.

DISCUSSION

MEMBER OF THE AUDIENCE: You talked of motivating the programmers. Have you been at all successful in actually motivating them?

MOUTON: I find that it takes more than motivation; it takes a little bit of control on the scheduling. When programmers operate in a very hectic atmosphere, it is very difficult. You can motivate and train them to do a particular job and get them really interested in doing it, but when another job comes across the desk that should have been done yesterday, then you are almost always forced to document after the fact. Where I have seen documentation done properly, it was a requirement. The documentation was done before the job could be put in an operational status.

MEMBER OF THE AUDIENCE: How was that requirement imposed?

MOUTON: The jobs came into the programming shop in the form of an analysis document or something like that and what was to be done had to be specified. The programming branch then developed the program that was turned over to an operational group to run. So before the job was turned over to another group to run, you had to have not just the operating instructions but the complete document. This operation was for the Army Map Service, and because it was sent out of the installation for safekeeping, all of this was done systematically. The time for documentation was included in the original estimate.