Improving software maintenance attitudes

by PAUL C. TINNIRELLO
The A. M. Best Company
Oldwick, New Jersey

ABSTRACT

Attitudes towards maintenance have been an overlooked source of problems in the software maintenance process. In the past, there has been little recognition of the significance of how attitudes affect the performance of maintenance functions. Investigation into the origin of these attitudes has led the author to formulate feasible solutions that foster productive attitudes through the educational and professional work environments.
INTRODUCTION

Progress in the development of software maintenance techniques has been languid in comparison to the growth in software development procedures. While fourth generation software promises to ease the maintenance difficulties, it does not change the fact that maintenance today is performed on software that has been developed in the past 25 years. It is not surprising, then, to discover that as much as 80% of all software costs are spent in maintenance effort while only 20% of the cost is invested in developing systems that will possibly have software simplicity. The neglect in software maintenance development has placed a stigma on the maintenance process. In addition, there has been a serious impact on the performance of maintenance in the programming environment as a result of the attitudes arising from poor maintenance procedures.

DEFINING THE MAINTENANCE PROCESS AND IDENTIFYING THE PROBLEM AREAS

The software maintenance process can be interpreted as the correction, adaptation, and enhancement of computer programs and systems. This definition of the maintenance process is widely accepted among those in the data processing (DP) community. However, finding agreement on what constitutes maintenance problems has been a stumbling block for years. Part of the difficulty in defining the problems stems from the way software maintenance is viewed. Management may have a different concept of maintenance functions than a programmer who is directly involved with maintenance activities. Still another viewpoint may come from the end-user who has extracted a notion of maintenance from both management and programmers.

It is the opinion of the author that software maintenance problems can be segmented into three areas:

- maintenance management,
- maintenance programming,
- maintenance attitudes.

Maintenance management can be defined as the management of the software maintenance process within the computer-based organization. Maintenance management affects programmers, managers, and end-users, and requires the careful integration of all parties towards a successful solution, whether it be correction, adaptation, or enhancement. Maintenance programming can be defined as the technical methodology in which a correction, adaptation, or enhancement occurs. Such methodologies include programming practices and techniques implemented within existing software systems. Finally, maintenance attitudes can be defined as the position an individual has towards the software maintenance process in its entirety. Maintenance attitudes are usually held by many members of the computer-based organization, with the strongest attitudes being held by those who have the greatest interaction with the maintenance process.

INTERDEPENDENCIES IN THE MAINTENANCE SOLUTION

If there is to be any development in the software maintenance process, then each of the problem areas of maintenance management, maintenance programming, and maintenance attitudes must be improved. Recognizing this fact is, of course, much easier than agreeing on which area has the most impact on the maintenance process. Thus far, it appears as if the emphasis has been placed on the maintenance programming category. One of the approaches used in this area has been the evolution of structured programming techniques, which promised program maintainability through a modifiable and adaptable design. Academic institutions, especially DP organizations, began to stress the use of structured programming techniques with the naive hope that maintenance complexities would eventually be eliminated. Unfortunately, structured concepts have not eradicated all of the maintenance problems. They have eased, however, some of the complexities in the maintenance process. In addition, attitudes in performing maintenance functions have improved for those individuals who are responsible for maintaining structured programs and systems. This improvement in attitude, which was elevated by the improvement in programming technique, demonstrates how each maintenance area is dependent on the other for success. The converse is also true. Poor maintenance management would affect the quality of work being performed and also diminish the attitude toward the maintenance process.

At this point, the author would like to suggest that maintenance attitude is a problem segment that, when improved, can have more benefits for the maintenance process than improvements in the other problem areas. Until now, maintenance attitudes have been recognized only as a result of changes in the maintenance-management and maintenance-programming segments.

COMPLEXITY OF MAINTENANCE ATTITUDES

The dynamics of attitudes is not fully understood. Attitudes can be a mixture of emotional and mental processes that an
ATTITUDE DEVELOPMENT

In an attempt to find a method for improving maintenance attitudes, it is necessary to uncover the origin of such attitudes. First, there is a need to examine those attitudes that grow out of the work experience. The majority of programmers usually encounter maintenance duties during the first several years of their professional careers. In some cases the maintenance work may be moderate to light, while in others the maintenance responsibilities can be heavy. Some organizations have a definite policy with respect to new programmers that requires that they be assigned maintenance responsibilities in order to better their understanding of existing systems and to improve their software skills. This philosophy has been noted for its advantages by advocates of the software maintenance process. However, such exposure to maintenance, especially with new programmers, can be detrimental to the organization and can possibly cause the development of poor attitudes about the maintenance process. The point can be argued either way about maintenance benefits, but the net result in attitude is usually negative, even though some valuable experience was gained.

Perhaps this suggests that it is unpopular to have a good attitude toward maintenance work. In any case, the effects of maintenance have been recognized by programmers involved with maintenance-related activities. Such effects include high turnover, low productivity, and excessive software costs. Programmers who reflect the effects of the maintenance process often possess certain attitudes about their work. These high-level attitudes, as they will be termed, include boredom, defeatism, frustration, and a feeling of lack of recognition. They are usually attributed to some set of conditions that is met while the programmers perform maintenance functions. These conditions or high-level factors, as they will be termed, include poor, little or no documentation; unstructured or poorly designed programs and systems; poor programming practices such as excess switches, meaningless data names, and nonstandard language commands; and extra work hours, odd work hours and pressure to complete maintenance tasks in little or no time. There is no doubt that the outgrowth of these high-level factors results in a poor or negative maintenance attitude. More important is the possibility that high-level factors, which foster high-level attitudes, also perpetuate the high-level factors.

As an example, consider a programmer who is given the task of performing a maintenance function on a poorly designed, poorly documented old program that has been the maintenance responsibility of twenty-five prior programmers. The probable result of such a task for the average programmer is that an attitude is either developed or supported against the maintenance process. In turn, the programmer will probably not provide any more insight into the program than was originally given. The author's experience has shown that the programmer's attitude will allow only minimal documentation and programming techniques to be performed, perhaps even burdening the program with poorer code. While this example might be overemphasized, it does illustrate how an attitude perpetuated more software maintenance complexity. High-level factors exist in many computer-based organizations and it is not likely that they will immediately disappear. It is safe to say, however, that attitudes developed in this environment need to be rectified if the maintenance process is to improve.

Another source of maintenance attitudes comes from a more fundamental area than the work place. These are the attitudes that grow out of the educational experience, and their roots lie deep within our educational behavioral patterns. Two of the important individual needs developed during this experience are those of creativity and skill growth. Much of the initial exposure to computer programming was through educational experiences that permitted the creation of programs as a method to learn new skills. As a result, little or no encounter with the maintenance process occurred except for the individual program debugging (and that task was consciously justified as part of the development scenario). Attitudes towards maintenance were not even realized at this stage. However, the attitudes that supported the theory that programming is a creative and skill-strengthening process flourished. Perhaps the occupational title of programmer as opposed to software engineer connotes the creative attitude as well.

When new programmers are exposed to real software maintenance situations, they are totally unprepared to handle the depth of the software maintenance process. They find themselves performing a programming task that is constrained by another style as well as design. In addition, they are confronted with the responsibility of understanding a program whose functions may be totally unfamiliar to them. It has been argued, however, that it is very possible that maintenance functions will provide new skill growth with programs that employ current software techniques. Unfortunately, programmers are usually assigned maintenance on specific systems for some duration and in time they will achieve the maximum skill growth that can be derived from such a system. In addition, not every system employs new or current software techniques. Therefore the time it takes to outgrow the skills of the system may be very short.

The need for creativity and skill growth extends beyond the educational environment into the professional work place. When these individual needs are denied, the result is usually a search for a place where they can exist. Recall that one of the effects of software maintenance is high turnover. It is from
these concepts that the author suggests that a fundamental problem in the software maintenance process is the possible hindrance of creativity and professional skill growth. This hindrance may create low-level attitudes, as they will be termed, which include uncreativity and nongrowth. It is also possible that these low-level attitudes create a subconscious attitude in the programmer that manifests itself in the more recognizable high-level attitudes described earlier.

Recognition of where maintenance attitudes originate points towards a method of how they can be improved. Since the attitudes described in this paper stem from the professional work place and the educational environment, it is only natural that a method of improvement occur in these places. The author has provided suggestions for maintenance attitude improvement via the conventional educational techniques currently in use. These suggestions do not exclude the fact that there are probably other techniques available. However, the author would like to stress the fact that the source of any method for improvement must come from the professional work place and the educational environment.

SOLUTIONS IN THE EDUCATIONAL ENVIRONMENT

Formal education curriculums, which include programming courses, should also include courses that address software maintenance issues. There is also an important need to clarify what the occupational functions of a programmer entail. Secondary schools that offer programming classes to students might want to structure their course objectives to include exposure to the maintenance process. At this educational level it is not necessary to investigate the methodologies used in maintenance but rather to introduce the concepts of what maintenance is about.

At the college and university levels, students who are required to take data processing courses as part of a non-DP curriculum should also be exposed to the maintenance issues. As potential users of computer-based systems, they would develop attitudes that may eventually be part of a computer-based organization. If nothing else, the user community would at least be conscious of the complexities involved in the maintenance process.

Finally, and most important, are those college, university, and programming schools whose curriculums are designed for computer science and programming graduates. A software maintenance course or courses should be required as part of the requirements for graduation. The content of such a course can be divided into the three areas of maintenance management, maintenance programming, and maintenance attitudes. One of the major objectives of the course is the realization that software maintenance is essential for the success of the computer-based organization. It would be of little value to present the course in the way maintenance has been viewed in the past, that is, “It’s a necessary evil.”

The particular topics of a software maintenance course may include:

- the need for maintenance,
- what are maintenance functions,
- tools for maintenance,
- preventive maintenance,
- planning maintenance groups,
- interfacing with users and developers,
- change control procedures,
- monitoring maintenance activities,
- attitude strategy,
- integrating creativity,
- maintenance programming practices.

Many of these topics would seem theoretical to students who have never been involved in a computer-based organization. However, a very practical topic, which can almost be the course in itself, is that of maintenance programming practices. Choosing a language that is a curriculum requirement, an instructor can easily create assignments where students need to correct, adapt, or enhance a prewritten program or programs. In the correction assignment, students are given the task to find and correct a problem that was the result of a poor design or oversight in functional specifications. Two programs, one structured, the other unstructured, would be given to the students. The grade in such an assignment could be dependent on the success of corrective action, or method of implementation, that is, maintaining program design uniformity, documentation standards, and the time it would take to make the change. Of course, it may take several assignments before the time factor would be meaningful. In the adaptation and enhancement assignments, the instructor could use the corrective assignment programs and ask for implementation of new functions or modify some existing functions. Use of consistent programs might strengthen the maintenance concepts without frustrating the student, a possible side effect in such a course. If frustration becomes a barrier to completion of an assignment, then perhaps the instructor could alter the assignment to allow the student to find a technique for getting around the frustration.

Another topic area is the integration of creativity and skill growth within the maintenance process. Here students could learn how to channel creative energy in a direction that is beneficial to both the maintenance task and the individual need. Typically, this topic would concentrate on the enhancement aspects of the maintenance process.

Another course for software maintenance might be a study of maintenance programming techniques for specific program languages, for example, FORTRAN, COBOL, PL/1, and ASSEMBLY. In this course the concentration would be on the problems and solutions for correction, adaptation, and enhancement within the constraints of the language. An assignment that an instructor could give might be to modify two different language programs to perform a similar function. Again, many assignments can be created to support such a course objective. If a programming-language maintenance course is not practical, then another alternative would be to include a maintenance section within the teaching of formal languages themselves. In such courses, students can learn the advantages of a programming language in the development process as well as the technique to be used with the language during the maintenance process.

As of this writing, the author has found little or no evidence
that indicates that higher educational institutions are requiring maintenance courses as part of a computer science degree program. Maintenance courses, like those suggested, would help current and future programmers by raising their consciousness of the maintenance process and by improving their maintenance attitudes. It is hoped that the shock of maintenance functions, which new programmers encounter, will thereby disappear.

SOLUTIONS IN THE PROFESSIONAL WORKPLACE

Improving maintenance attitudes in the workplace will be more difficult than resolution in the educational environment. Unfortunately, many programmers have less than positive attitudes already established. However, changing attitudes is definitely possible. One of the best ways to initiate an improvement is to have the computer-based organization demonstrate a positive recognition of the maintenance process. Programmers and managers involved with maintenance activities could be given extra incentives for their work. Such incentives could include extra compensation while assigned to maintenance groups, time compensation for extra or odd work hours, and advanced training in software techniques.\(^3\) Another improvement method is the establishment of a planned development path for programmers. A typical path may include the rotation of maintenance assignments and development assignments on a regular schedule. This will allow programmers the opportunity to flex their creative skills and to support their need for professional growth. There is much evidence that indicates some computer-based organizations are already gearing up for some type of attitude improvement strategy. One of these strategies is the use of outside training seminars in software maintenance for managers, project leaders, and group leaders.

Aside from commitments on the part of the computer-based organization, there is a need for individuals to examine their own methods for improving attitudes. Computer science is a technology that almost dictates change, and it is vital that individuals be conscious of those changes that are needed for professional development.\(^10\) Professional programmers must commit themselves to improving attitudes about maintenance. Given the implementation of maintenance courses in the formal educational environment, programmers can attend those classes to improve maintenance skills and attitudes. After all, there seems to be that kind of commitment whenever a new software technique is introduced.

Attitude development in the professional workplace is possible through a mutual effort on the part of the programmer and the computer-based organization. Perhaps it will require the educational community to react first before individuals and companies commit themselves to an attitude improvement plan.

CONCLUSION

Focusing the effort on improving attitudes towards maintenance will help in the development of the software maintenance process. Attitudes are complex, but their complexities can be more easily understood if the origins of these attitudes are examined. Recognizing the source of attitudes will foster new methodologies in attitude improvement that can thus become part of the programming process. No single solution to attitudes will result in attitude change. It will take time for attitude improvement theories to become acceptable as solutions to maintenance problems. The educational environment, which enabled the achievement of new software techniques, will also be inspirational in the development of new maintenance concepts. The future of software maintenance looks more promising with the recognition of underlying problems and implementation of new solutions.

REFERENCES