Human investment techniques for effective software maintenance

by NICHOLAS L. MARSELOS
Western Electric Company
Lisle, Illinois

ABSTRACT

This paper presents methods for improving the maintenance of software by addressing the psychological issues that impact on software maintenance personnel. The emphasis in this paper is on making software maintenance developers more effective through goal setting, by using team-building approaches, through support personnel, and by using skill profiles to plan for their technical growth.
INTRODUCTION

Today everyone recognizes the problems in software maintenance. Over half of the people now developing software are involved in maintaining it. This absorbs a great deal of the energy and creativity that we have in our software-development community. With increasing emphasis the question is being asked, “What can be done to solve this problem?”

Unfortunately, the difficulties in software maintenance are not the result of a single problem, but of many. Many single solutions have been employed to make software-maintenance developers more effective. Most of these have been technical solutions that help the software-maintenance developer design or program or control the software product more effectively. There has, however, been a real lack of emphasis on the human investment in software maintenance. Although the technical solutions are beneficial and important, the real gains in improving the productivity and effectiveness of software-maintenance developers are attained by improving their motivation. This paper focuses on the psychological issues that plague software-maintenance developers. It offers multifaceted solutions that, when applied, will improve their motivation and provide many other benefits to their companies.

MAINTENANCE IS A MULTIFACETED ACTIVITY

The technical nature of software maintenance is a well-understood problem. The software-maintenance developer performs a variety of activities. These include defect correction, feature addition, and working with users. The software maintenance developer must also keep current on the system and support environment in which his software product operates in order to change the product as required by changes made in the environment.

The software maintenance developer must constantly use some of his creative energy to understand and get around the constraints of the software product he or she is maintaining. This product is usually poorly documented and in many cases written in an unstructured and very difficult to change manner. The software maintenance developer must also spend part of his or her time dealing with the user in a variety of roles that may include trainer, consultant, and complaint handler. All of this is usually done with time pressures resulting from very short development schedules.

These parts of the software-maintenance developer’s job are well understood, and many solutions have been offered to help in these areas. But there is an entire area in which very few solutions have been offered, and which creates more significant problems for the software-maintenance developer.

Psychological Issues of Software Maintenance

The software-maintenance developer is in a position with a great deal of psychological pressure. This pressure comes from a variety of sources. One source is the feeling of being “stuck” in the job of maintaining a specific software product. Often developers feel that management is indifferent to their problems. The management passes down the message “Don’t make waves, just get the job done.” The management’s attitude may be reflected in their lack of interest in the education and personal growth of the software-maintenance developers. This makes software-maintenance developers feel like second-class citizens in the organization.

The evolution of software maintenance has resulted in an environment of independent islands. Each island supports one or a few software-maintenance developers working on their specific product, unattached and uninvolved with much of the rest of the development organization. In this environment, the software-maintenance developer feels unsupported and that the job rests solely on his or her shoulders.

The psychological pressures on software-maintenance developers has a demotivating effect. Over a period of time, their productivity begins to diminish and this has bad effects on the cost and quality of the software products being maintained.

Administrative Problems

The administrative problems of software maintenance as seen by management revolve around the issue of keeping cost to a minimum. This cost directly correlates with the number of people involved in maintaining the software products. Management is faced with the problem of optimizing to the minimum number of software-maintenance personnel that can keep software products maintained in a healthy way. However, this often creates the alienation and morale-lowering feelings experienced by software-maintenance developers.

The rotation of software-maintenance developers to different assignments and the ability to provide backup for them is often a serious administrative problem. This problem stems from the long-learning curve required by the software maintenance developer to effectively maintain the software product. The lack of the effective documentation required by the software maintenance developer to effectively maintain the software product. The administrative problems of software maintenance as seen by management revolve around the issue of keeping cost to a minimum. This cost directly correlates with the number of people involved in maintaining the software products. Management is faced with the problem of optimizing to the minimum number of software-maintenance personnel that can keep software products maintained in a healthy way. However, this often creates the alienation and morale-lowering feelings experienced by software-maintenance developers.

The rotation of software-maintenance developers to different assignments and the ability to provide backup for them is often a serious administrative problem. This problem stems from the long-learning curve required by the software maintenance developer to effectively maintain the software product. The lack of the effective documentation required by the software maintenance developer to effectively maintain the software product. The administrative problems of software maintenance as seen by management revolve around the issue of keeping cost to a minimum. This cost directly correlates with the number of people involved in maintaining the software products. Management is faced with the problem of optimizing to the minimum number of software-maintenance personnel that can keep software products maintained in a healthy way. However, this often creates the alienation and morale-lowering feelings experienced by software-maintenance developers.
MULTIFACETED SOLUTIONS TO THE MAINTENANCE PROBLEMS

The overall effect of the administrative problems in software maintenance is to demoralize the software-maintenance developer and to frustrate management. These problems have a negative impact on the software product, which over a period of time begins to degenerate. Software management sees a large investment slowly dwindling away with little control on their part to effectively reverse the process. The software-maintenance developers feel pressed in the position of too little support and too few resources to do the job effectively.

A key factor in improving the software-maintenance developers’ productivity and quality rests in motivating those developers. An environment must be established in which the software-maintenance developers feel supported and can proceed to do their job in the most effective way. This takes a commitment by management. It also requires a cultural change that is first initiated by defining the organizational objectives, and then enhanced by creating a supportive environment that meets the psychological needs of the software-maintenance developers.

Goal-Setting for the Organization

One of the most important needs in any organization is to have well-defined goals in which everyone in the organization supports. Software maintenance has its own set of goals. These goals can conflict with each other, as indicated in the experiment by Weinberg.1 Table I identifies the goals and the negative effects of those goals.

Software-maintenance goals must be set, then ranked to avoid conflicting goals. As Table I shows, setting the goal of timeliness, that is getting the products out on time, may jeopardize the efficiency and maintainability of the software products being developed. This is true whether the products are in the initial development stage or in the maintenance-development stage of their life cycle. Table I also shows that an organization both placing emphasis on timeliness and demanding high maintainability puts tremendous pressure on the software-maintenance developer, since these two goals are in direct conflict with each other. It is therefore important not only to establish the goal or goals that the organization would like to achieve but also to rank those goals to minimize conflicts.

The process of establishing goals for the organization has to begin with management. The management should sit down and decide what goals they really seek, whether it is to have customer satisfaction or improved maintainability or timeliness of its products. The goals may be global to the organization or modified for each project. Once the goals are selected, they should be ordered in terms of priority, the most important, the next most important, and so on. Once this list is created, it should be checked to make sure that conflicting items aren’t adjacent. If they are, the management must then decide which is the most critical item of the two and move the conflicting item down on the priority list. Finally, the entire management must support these goals.

After the goals are well defined and written, everyone in the organization must be informed what the goals are, and their support must be promoted. This can be done by having meetings of management and the software-maintenance developers. These meetings should be used to stress the importance of these goals to the organization and to encourage suggestions on how best to achieve the goals.

The success with which these goals will be accepted depends on whether achievement of these goals is rewarded. The rewards can be of two forms. First, there must be recognition awards. These are publicized awards for software-maintenance developers who achieve the goals established by the organization. For example, if timeliness is a specific goal, then those developers that bring their products out on time would receive recognition for their accomplishments either in the organization’s news bulletin or in memoranda. Second, there should be substantive awards given for achievement of the goals. These include pay raises and promotions or any other form of substantive award. In all cases, it should be made common knowledge that the award has direct correlation with achievement of the goals set by the organization.

When goals are established and promoted in such a way, they will have a profound effect on the cultural aspects of the organization. The software-maintenance developers will be more motivated once they have clearly in mind what the organization considers important in the performance of their activities. This will give the software-maintenance developers objectives to direct their energies toward. It will not, however, set up the type of supportive environment that is necessary for them to be effective in achieving their goals. For that, a more supportive team environment must be created.

### Table I—Goals of Software Maintenance

<table>
<thead>
<tr>
<th>Goal</th>
<th>Negative Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeliness</td>
<td>Inefficiency in operation</td>
</tr>
<tr>
<td>Operational efficiency</td>
<td>Increased development costs</td>
</tr>
<tr>
<td>Customer satisfaction</td>
<td>May not be user friendly</td>
</tr>
<tr>
<td>Minimum costs</td>
<td>Development longer</td>
</tr>
<tr>
<td>Improve maintainability</td>
<td>Greater resources required</td>
</tr>
<tr>
<td></td>
<td>Schedules shortened</td>
</tr>
<tr>
<td></td>
<td>Better project planning required</td>
</tr>
<tr>
<td></td>
<td>User satisfaction can suffer</td>
</tr>
<tr>
<td></td>
<td>Schedules longer</td>
</tr>
<tr>
<td></td>
<td>More resources required</td>
</tr>
<tr>
<td></td>
<td>More emphasis on documentation</td>
</tr>
<tr>
<td></td>
<td>More emphasis on design</td>
</tr>
</tbody>
</table>

Instituting Team Consciousness

Most software-maintenance developers are put in a situation where they have few people to rely on for support. New project developments often have several team members working in concert to complete the project. When the project is in its maintenance cycle, only a skeleton force, usually just a single individual, will maintain the project. The software-maintenance developer is left without having the benefit of sounding boards or other people’s expertise to resolve prob-
Augment Groups

Augment groups are informal teams chartered to propose solutions to problems or to suggest improvements for the organization. They are founded on the concepts employed in quality circles now popular in Japan. They differ from quality circles in that they are populated by professionals and that they can apply to any specific goal that the organization seeks.

The charter of an augment group is implied by its name. The name and charter of the group should be chosen so as to be directed to a specific organizational goal. For example, there could be productivity groups or user-relationship groups or more-effective-documentation groups. Each group would be chartered to look into problems relating to one specific area.

The process of organizing and conducting these groups is now well defined in the literature pertaining to quality circles. The groups should be formed on a volunteer basis. Software-maintenance developers should join the groups that most interest them. Several groups can be established within the organization. In the case of multiple groups, a facilitator or coordinator should attend the group meetings to ensure that there is no significant overlap in the activities or the solutions of the groups.

Augment groups in the area of software maintenance have to perform two vital functions. First, they offer an opportunity for some of the organization's problems to be addressed in a creative fashion; second, they give the software-maintenance developers an opportunity to participate in a team effort. The first benefit is one for the entire organization. Very often augment groups suggest substantial improvements that save the organization money or improve the productivity or quality of its products. The second benefit of augment groups is to provide a vehicle for the exchange of information among software-maintenance developers. This gives them a feeling of team involvement even though their normal daily job might isolate them.

Peer Review Units

Two of the major problems facing software-maintenance developers are the lack of support in ensuring that a product they have developed is of good quality and will work effectively and second, the lack of backup personnel to relieve the load in pressure situations. Both of these problems can be relieved by the use of peer review units. A peer review unit is a group designated to review the work of the members within that group. The product can be the design documentation, the analysis documentation, or the program code itself. The group has a permanent membership. The membership is chosen to achieve the compatibility conducive to fostering "ego-less," or defensiveless, participation as described by Weinberg. The review process could be conducted as a structured walkthrough. The review should have some formality, as described by Yourdon. At the minimum, it should include a signoff sheet to indicate that the reviewers have accepted the product being reviewed.

The value of peer review units is in the improved quality of the program products being developed. The units also provide a backup situation by having the reviewers learn about the different systems that are involved. The results of field trials indicate that peer ratings of programs can be productively nonthreatening, and serve as incentives for programmers to produce higher quality code.

Application Area Groups

Another type of group that can help in the communication and information and the support of software-maintenance developers is the application-area group. These are groups such as user-interface groups, or maintenance-developer groups, or birds-of-a-feather-type groups that work on problems or exchange information that is of common interest to the group. The group meets periodically to discuss either common problems or solutions, or to present new features or ideas, or to be used as a sounding board for activities or plans within the organization.

These types of groups are the easiest for an organization to establish. They are conducted as simple and informal sessions. They offer the software-maintenance developers an opportunity to exchange information and experiences. If users are involved, it provides a mechanism in which the software-maintenance developers and the users can improve their relations by gaining a better understanding of each others' problems and points of view.

Improving the Effectiveness of the Maintenance Personnel

The most significant qualification of software-maintenance personnel is that they have the technical capability to perform their job. This requires a well-planned training program. Another very important aspect of training is that the software-maintenance developer feel they are personally growing in their technical expertise and are keeping up with the state of the art of their profession; (the substantially high need for growth by software professionals was shown by Cougar and Zanack). Both of these are also important to the organization. Obviously, if the software-maintenance developers cannot cope with the technical aspects of their job, the product they are maintaining will suffer. If on the other hand, the software-maintenance developers have a strong desire to grow technically and that desire isn't met by an effective program for personal growth, then their morale becomes low and their productivity is negatively affected.

The training of software-maintenance developers must be a well-planned and controlled activity. The first step is to understand the skills that the software maintenance developers already have. This can be done by developing a profile of the skills they have acquired through education and on-the-job
experiences. Next, the organization should profile the skills necessary for various jobs. Finally, the organization should profile the educational or training vehicles that can teach those skills. With this information, it's possible to develop an educational plan. This plan can be used for the rotation of software-maintenance personnel and also for planning their personal growth and training in a direction that benefits them and is suitable for the organization's needs.

Support Personnel

Many of the activities of the software-maintenance developer are of a low-level clerical nature. These are repetitive and time-consuming activities that diminish the software-maintenance developers' overall effectiveness. Many of these jobs, however, can be delegated to support personnel.

Software support personnel benefit the software-maintenance developer. They perform functions at various levels depending on their own abilities and expertise. These can range from simple data-entry jobs to performing the software testing. A more skilled helper can provide the first-line interface with users by handling some of the simple operations and questions needed to support users.

The organization will also benefit from having software support personnel. First, it reduces the cost of software maintenance by making the software-maintenance developers more effective. It allows the software maintenance developers to have more time for their specific activities and relegates the clerical support work to a lower-salaried individual. The presence of software support personnel also enforces a certain level of documentation: the documentation needed to help them do their job. Second, they provide continuity for the organization when new software-maintenance developers begin to maintain the product. Finally, a software assistant can be shared by several software-maintenance developers, providing added cost savings to the organization.

Software-maintenance developers can also be helped by expert technical support. This can be instituted by circulating a list of the persons in the organization who are most knowledgeable about specific areas, for example, job-control language or program languages. The function of technical support experts is to answer inquiries about specific problems in their particular areas of expertise. This can be done by setting up expert tables where maybe for an hour or two every day the expert would sit and field all questions.

The use of technical experts within an organization works if the organization actively supports the policy of getting the job done in the most expeditious and effective way. This means that the organization must discourage the not-invented-here syndrome and must encourage software-maintenance developers to use their innovativeness to meet the organizations' stated goals and not simply waste their energies on problems where they lack expertise. Rewards must be established for both the technical support person and for those who seek his or her help.

CONCLUSION

Software-maintenance developers often find themselves in an unrewarding and stress-filled job. They may feel ignored by their management and alienated from the rest of the organization. They may suffer from not having a full understanding of the goals of the organization and from the lack of support personnel to help them do their job effectively. They may have the feeling that they are stuck in their jobs and limited in their professional growth.

This paper describes several approaches that can be applied to making software maintenance developers more effective. The approaches deal with the psychological aspects of software-maintenance developers. The emphasis is on making them feel more a part of the organization and giving them more effective support. By using these approaches, the software-maintenance developer's motivation can be improved remarkably. Only when this is done is it possible to gain the maximum benefits from the technical tools and techniques for improving software maintenance.

REFERENCES