# Software Maintenance: A Budgeting Dilemma

by Dan Hocking and Joe Celko

Army Institute for Research in Management,
Information and Computer Science

SUMMARY: There is considerable effort to reduce the software budget devoted to the maintenance of applications systems. This effort will have the effect of improving productivity of development and maintenance programmers. This means that for a given system over a given time period, the amount spent on software maintenance can be reduced significantly. The reduction might even reach the eighty per cent sometimes shown in the literature. We support and applaud the efforts to improve software maintenance procedures. Despite this type of reduction, it is not certain that organizations will spend less on maintenance relative to development. It is likely that the opposite will occur as more systems are being supported. This paper shows how that can happen through the derivation of some simple cost equations.

KEYWORDS: software maintenance, budget, project management

INTRODUCTION: Systems tend to behave in a counter intuitive fashion. When a direct solution is applied to a problem, the result can be the opposite of the result intended by the people who applied the solution. There is a belief that the high cost of software maintenance can be reduced by improving the quality of software. It seems reasonable that if every piece of software is easier and cheaper to maintain, then the funds budgeted for software maintenance can be reduced. There is much effort to reduce the costs of data processing in the Federal Government and specifically within DoD (e.g. Ada). At the 1983 Federal Software Conference, sixty seven percent of software costs were reported to be software maintenance.[SOR83] In another session, it was reported that structured programming practices can reduce software maintenance costs by eighty percent. The proceedings did not identify the budgetary implications of such an improvement. Let us inspect the premise, granting a whole set of optimistic and simplifying assumptions, and see what really happens.

BACKGROUND: James Martin and Carma McClure have published a book titled "Software Maintenance: The Problem and its Solutions."[MAR83] In that book they portray software maintenance as something to be solved and indicate that some organizations have reduced their time spent on software maintenance from eighty percent of the time to twenty percent of their time. That book offers several constructive approaches to reducing software maintenance. We applaud these efforts and support them. We believe, however, that the Data Processing budget will continue to be heavily oriented toward software maintenance and that to promise otherwise is being unduly optimistic. The purpose of this paper is to demonstrate that the dramatic improvements in software maintenance will not necessarily lead to dramatic shifts from software maintenance activities to software development activities for the data processing staff. Even if such a shift is accomplished initially, in the absence of continuing improvements in software maintenance procedures or in continuing increase of programming resources, the steady state division of resources between software development and software maintenance will be to devote one hundred per cent of all programming resources to software maintenance.

We accept the premise of Parikh & Zvegintzov that "Maintenance is an essential element in the life of a software system."[PAR83] Furthermore, we are using their definition of software maintenance which is "Software maintenance is the work done on a software system after it becomes operational." Specifically this includes the activities that Martin and McClure call Corrective Maintenance, Adaptive Maintenance, and Enhancement Maintenance.-[MAR83] We would even include the Support activities that Martin and McClure identify separately.

With the above background we do believe that for a given system you can reduce the annual cost of maintenance dramatically. If, for example, a system is maintained traditionally and you must have two programmers working full time, you will have reduced software maintenance by fifty per cent if by using new technology you need only one programmer working full time. If the operational life of the system is doubled, then for the life of the system, total maintenance costs are the same even though annual costs of maintenance have been reduced fifty per cent.

For the organization, improved software maintenance procedures free up resources for new development. As new systems are developed and placed into operation, more maintenance resources are required. Two systems requiring one programmer per year cost the same as one system requiring two programmers per year. It is clear that productivity has been improved with the improved maintenance procedures even if total maintenance costs have not been reduced and the percentage of costs devoted to software maintenance has not been reduced.

DERIVATION OF COST EQUATIONS:

We will establish some simple cost equations to discuss the effect of programming practices on costs. The first equation is:

(1) TC = DC + MC

TC is the total cost of a particular system in personnel months. DC is the development cost for the system in personnel months. MC is the maintenance cost for the system in personnel months. The expression for development costs can be expanded as:

(2) DC = DP * DT

DP is the average number of programmers involved in the development of a system. DT is the number of months in the development phase of the project. This expression could be replaced with a summation of cost for each month of the development, but let us make life as simple as possible. In a similar way, the expression for maintenance costs can be expressed as:

(3) MC = MP * MT

MP is the average number of programmers involved in maintenance. MT is the number of months in the operational life of the system. There are other costs involved, but again we are simplifying. With the expansions of equations (2) and (3), equation (1) becomes:

(4) TC = (DP * DT) + (MP * MT)

If we change software practices, then we can affect any of the four terms. That can be represented as:

(5) TC = ((a*DP)*(b*DT)) + ((c*MP)*(d*MT))

The terms a, b, c, and d represent the relationship of the various factors between the original and the new improved practices. The implication of improved software practices is that we can reduce the level of effort required to develop a system (factor a). We assume that this will be a desirable result. We might also reduce the amount of elapsed time required for developing a system (factor b). That also is assumed to be a desirable result. We may also reduce the level of effort required for maintaining a system (factor c). We assume that that would be a desirable result. The change in the remaining term d must, however, represent an increase since that is what would be considered desirable. In equation (5), then, it is desirable for a, b, and c to all be less than one, while it is desirable for d to be greater than one.

RATIO OF MAINTENANCE COSTS TO TOTAL COSTS:

It is time now to change perspective. Since we are interested in the proportion of expenses that are required for maintenance, we can set up an equivalent expression for that term:

(6) PMC = MC/TC

In this expression, PMC stands for the percentage of costs devoted to maintenance, MC stands as before for Maintenance costs (in personnel months) and TC stands for Total Costs. We can expand this equation using equation 3 and equation 5 to express PMC in the same terms that we were using before. The resulting equation looks like:

(7)

$$PMC = \frac{((c * MP) * (d * MT))}{((a*DP) * (b*DT)) + ((c*MP) * (d*MT))}$$

Every term in this equation has the same meaning that it had before and we can rearrange it for our convenience and substitute back to obtain:

(8)

$$PMC = \frac{c * d * MC}{(a * b * DC) + (c * d * MC)}$$

SAMPLE COST RELATION DATA:

What figures should we use for those parameters? Let us pick the most optimistic ones published to date. The results of a Structured Programming Study by Infotech involving 1,000 companies worldwide was quoted at the 1983 Federal Software Conference.[SOR83] They claimed that Structured Programming can

a. Reduce project time (in programmer/months) up to 50%
b. Reduce implementation time up to 30%
c. Reduce program maintenance up to 80%

Having this data, we must now interpret what it means. In terms of the above equations for example, the reduction of program maintenance could be that PMC is

reduced from 67 per cent to 13.4 per cent. An alternative explanation is that it could be that MC is reduced and the product c*d is equal to 0.2. A third alternative interpretation is that d is equal to 0.2. It is our belief that the only valid interpretation is the last alternative.

## DISCUSSION OF ALTERNATE INTERPRETATIONS OF SAMPLE DATA

Let us examine the alternatives. In the first case, if we assume that development costs remain constant, then we have:

$$(9)\quad PMC = 0.134 = \frac{c * d * MC}{DC + c * d * MC}$$

For the moment, assume d = 1 since it is unlikely that a shorter operational life would be considered favorable. We can then solve for c.

$$(10)\quad c = \frac{0.134 * DC}{0.866 * MC} = 0.154 * DC / MC$$

With DC/MC currently equal to 0.5 the final value of c is 0.077. Such a change in the rate of maintenance work would surely have some effect on the length of the operational life. Any such effect means that d becomes greater than one and that c becomes even yet smaller. The effect has not been so measurable that this seems a valid interpretation especially when we also see the development costs are also reduced. Taken together, we find the initial interpretation that the reduction in maintenance means that the proportion of resources devoted to maintenance is reduced by eighty per cent to be an improbable interpretation.

The second alternative was that MC was reduced by eighty per cent. This would mean the product c*d would equal 0.2. In this case, assuming the operational life is not changed for the moment results in a value of c equal to 0.2. With this change in the rate of maintenance work applied to a system, we would expect a longer operational life. If we apply this expectation of a longer operational life, the value of c becomes smaller again. We reject this interpretation as being at variance with current experience.

Although it is not the focus of this paper we have a similar ambiguity with the data for project time. We can assume that the data represents the product of a*b or the factor a. Since we have no value for b in the referenced study, which does not clearly state the meaning, we will here assume that project time represents the factor a. Although the numbers will change if we make the alternative assumption it does not affect the main conclusion that we cannot expect improved programming practices to significantly change the proportion of people involved in software maintenance.

## EFFECTS OF CHANGES IN SOFTWARE DEVELOPMENT AND MAINTENANCE PRACTICES

The remainder of the paper will use the third interpretation to show the effect of improved programming practices on the proportion of people involved in software maintenance. We make the assumption that the above reductions in software development costs and software maintenance costs are accomplished at the same time. We can substitute into equation (5), using project time as the rate of resource expenditures during development, a = 0.50. We will use implementation time as the length of the development project, b = 0.70. And finally, we will use the figure for program maintenance as the rate maintenance resources are used, c = 0.20. Since no figures are given for the length of the operational life, that will be left as d. The result is that the total cost equation becomes:

(11)

$$TC = (0.50*DP)*(0.70*DT)+(0.20*MP)*(d*MT)$$

This reduces to:

$$(12)\quad TC = (0.35*DP*DT) + (0.20*d*MP*MT)$$

From the substitutions in equation (12) we can see that total personnel costs have been reduced so long as the increase in the operational life is not enough to overcome the decrease in the other factors. This is in line with expectations and can be labeled "good".

Let us now change focus again back to percentage relationships. If we use the same values for a, b, and c that we used before (0.5, 0.7, and 0.2 respectively) and multiply out the constants we have:

$$(13)\quad PMC = \frac{0.2 * d * MC}{(.35 * DC) + (0.2 * d * MC)}$$

Other data reported at the 1983 Federal Software Conference suggest that sixty seven percent (67%)[SOR83] of current software costs are maintenance costs, or

$$(14)\quad MC = 0.67 * TC$$

we then have:

$$(15)\quad PMC = \frac{(0.20 * d * 0.67 * TC)}{(0.35*0.33*TC)+(0.20*d*0.67*TC)}$$

by multiplying the constants together we can then obtain the expression

$$(16) \qquad PMC = \frac{(0.134 * d * TC)}{(0.1155 * TC) + (.134 * TC * d)}$$

Since the term TC appears in each part of the equation on the right, we can cancel it out and thus obtain:

$$(17) \qquad PMC = \frac{0.134 * d}{0.1155 + (0.134 * d)}$$

If we assume for the moment that there is no effect on the operational life of the software then d = 1 and equation (19) becomes:

$$(18) \qquad PMC = \frac{0.134}{0.1155 + 0.134} = 0.537$$

This means that about 54% of software costs are software maintenance under the new environment instead of the 67% currently experienced.

Using the values of a and c in equation 11 (.5 and .2) results in software development becoming two and one half times more expensive relative to software maintenance than under traditional practices. This would indicate a probable lengthening of the operational life of the system. If we assume optimistically that the operational life were to be doubled then equation (18) becomes:

$$(19) \qquad PMC = \frac{(0.134 * 2)}{0.1155 + (0.134 * 2)} = \frac{0.268}{0.3835} = 0.70$$

Thus, an eighty percent reduction in the rate of expense for software maintenance can result in a change in the proportion of software maintenance. That change, however, can be either a reduction to about fifty percent or a slight increase to about seventy percent. Neither of these figures results in anything near an eighty percent reduction in software maintenance resources at the budgetary level. Other examples do reduce the proportion of expenses devoted to software maintenance more dramatically than does this example, but the most extreme example assumes no change in development costs, project length or the operational life of the system. If we plug those figures into equation (16) we get:

$$(20) \qquad PMC = \frac{0.134}{0.33 + 0.134} = 0.288$$

This substitution gives a PMC of 28.8%. An eighty per cent reduction in software maintenance at the budgetary level means that of one hundred dollars currently spent on software, if sixty seven are now spent on maintenance, under the new circumstances, only 13.4 dollars would be spent on maintenance. In those circumstances, however, the percentage of software costs due to maintenance is still 28.8 per cent. Even this most favorable example does not result in a budgetary improvement as large as might be assumed, since this is a rate of expense 44% higher than a direct reading of the above figures would indicate. It also is not realistic, as are none of our examples, since all assumed values were extremes. The implied assumption that the results are comparable between systems is also unrealistic. The reason is that the size of the system is not involved in these cost equations.

The interesting result is the effect of the improved programming practices on the total environment. The backlog of new system requests is all too well known in the trade. Let us assume that the improved methods of program production will make it possible to fill more of that backlog. The same amount of effort will be spent per year, but, as we have seen, a greater percentage will be in maintenance. Therefore, more new systems will mean more staff doing maintenance programming on them. The counter intuitive result is that maintenance programming increases, instead of decreasing, as a total budget item.

An example of this effect follows. If we assume a constant development staff of twelve programmers that can simultaneously develop three systems over three years (average one per year), and an operational life of ten years with a maintenance staff of two programmers per system, the twelve programmers will supply enough systems to occupy twenty maintenance programmers. If under the improved practices they can develop six systems over two years (an average of three per year) and the new systems have an operational life of twenty years with a maintenance staff of two programmers for every five systems they will provide work for twenty four maintenance programmers. The same number of development programmers have had a six fold increase in productivity (granting our assumptions and constant size systems) while having a five fold increase in the productivity of the maintenance effort. Under those circumstances, the proportion of maintenance to total effort has changed from about sixty per cent to about sixty seven per cent.

By way of an analogy, consider automobiles. The cost of maintenance on an early hand built car was very high compared to the purchase price. But people owned few cars in those days, so the total cost was relatively small. There were no full time auto mechanics because no one could earn a living at it. Instead, the work was done by machinists who knew how to machine custom parts. When assembly line cars were produced the cost per unit went down considerably, and the availability of mass produced parts saved the high cost of custom machined parts. The use of mass produced parts also reduced the cost of automobile maintenance. This turned a semi-skilled, every day activity associated with automobile operation into a separately identified, skilled, periodic activity. On a societal level, automobile maintenance became a much bigger activity because of the increased use of the automobile.

CONCLUSION:

We have developed simple cost expressions for development and maintenance costs. We've used these expressions and data from published reports to show that the proportion of software costs devoted to maintenance will not necessarily change as a result of improved programming practices. Efforts to improve software maintenance are good. Until the backlog of activities to automate is exhausted, however, no reduction in total costs should be anticipated. Moreover, budget planners should not use the promise of more reliable and easier to change software to reduce the budget for their software maintenance staff and their activities.

References


1. [MAR83] James Martin and Carma McClure, Software Maintenance: The Problem and Its Solutions, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1983.

2. [PAR83] Girish Parikh and Nicholas Zvegintsov, Tutorial on Software Maintenance, IEEE Computer Society Press, Sivler Spring, Maryland, 1983.

3. [SOR83] Alfred R. Sorkowitz, Software Quality Assurance - A Practical Approach, U. S. Professional Development Institute, Fairfax Virginia, 1983.