Problems and Issues in Application Software Maintenance Management

PRASHANT PALVIA

THE UNIVERSITY OF MEMPHIS

AARON PATULA

UNIVERSITY OF MINNESOTA

JOHN NOSEK

TEMPLE UNIVERSITY

ABSTRACT

This article is a comprehensive reexamination, after over a decade, of various software maintenance issues and problems. A survey was conducted of information systems professionals using a comprehensive questionnaire. Among other issues, twenty-six specific problems were rated by these individuals. The findings indicate that maintenance problems are still pervasive, and can be classified into seven primary areas: programmer time availability, programmer effectiveness, operations reliability, user knowledge, product quality, hardware/software limitations, and user training. The article also proposes a contingency model for explaining the various maintenance issues. Several hypotheses are postulated on the basis of the model, and statistical support was found for several of the hypotheses. Finally, managerial implications are discussed and recommendations are made based on the findings.

INTRODUCTION

Over a decade ago, *Communications of the ACM* published an article describing problems in software maintenance based on a comprehensive study performed by Lientz and Swanson [9]. This study was an indepth examination of software maintenance in data processing departments, and is described in detail in [10]. In an earlier article, Lientz et al. [11] described salient characteristics of application software maintenance.

More than a decade later, software maintenance remains a thorny issue and continues to consume significant resources from corporate information systems (IS) department budgets. Estimates of the ratio of maintenance to total IS budget have consistently ranged between 50% to 80% [1,8]. Yet, maintenance suffers from organizational obstacles. It is typically accorded a second-class status, and development of new systems often takes priority over it [12,16]. As Kim and Westin [8] have pointed out, maintenance is not merely concerned with technical issues but also with the environment of the department and organizational issues. Research into organizational aspects of maintenance is limited. Some suggestions offered for organizational alignment of maintenance include the redefinition of IS activities into "installed" and "future" systems [16], and having a seamless split between initial development and subsequent refinement [15]. Another suggestion offered is to make effective use of incomplete data for maintenance management [14]. On the technical side, Banker et al. [1] have examined the productivity of software maintainers and have studied the impact of size and complexity of business systems, and a number of environmental variables. Some of these factors may be manipulated to improve maintenance productivity.

Given the persistent problems in software maintenance, we reexamine, in detail, the current state and problems of maintenance. This article provides this new information as well as develops insights into current problems and practices in maintenance management. The same comprehensive questionnaire that was used by Lientz and Swanson [9,10] was used in the present study and was administered to IS professionals. Minor modifications to the questionnaire were made only to accommodate changes in software development methods that have since taken place. While summary remarks

comparing the Lientz and Swanson results and the present results are made in the article, the main focus is to present contemporary results and obtain insights into the many relationships.

THE SURVEY

The questionnaire used in this study is an adaptation of the one used by Lientz and Swanson [9,10]. The questionnaire is quite extensive and comprehensive (a total of 16 pages). It had the following sections: a section on demographic information about the organization, a section on general information about the MIS department, and several sections on a specific operational application system chosen by the respondent. The sections on the application system included: information about the various characteristics (e.g., size, programming language, database capability, system type) of the application system, types of maintenance activities being performed, types of maintenance controls used, and types of maintenance problems being experienced. The questionnaire was mailed to two hundred and forty members of the Data Processing Management Association (DPMA) members in the northeast and midsouth regions of the United States. Followups were conducted by both telephone and mail. A total of 52 questionnaires were returned providing a response rate of 22%. This response rate is typical of surveys reported in the Management Information Systems literature.

Some demographic information about the sample is worthwhile to examine. The average and the median DP budget of the respondents' organizations was in the range of $500,000 to $1 million. The direct manager of the head of the IS department in 39.6% of the cases was the CEO, and in 30.2% of the cases was the VP of Finance. In 69.8% cases, the participants worked in centralized information systems. 18.9% of the systems were decentralized. The median number of all employees in responding organizations was 550. The median size of the full time DP staff was 10 with 6 devoted to applications development and maintenance. The DP budget, on the average, was 3% of the total company budget. The median age of the application system was 38 months, and the average was 66 months.

MAINTENANCE PROBLEMS

Lientz and Swanson [9] reported a large set of maintenance related problems. This set of problems provides a fairly accurate and comprehensive representation of the nature of maintenance issues. The same set of 26 problems was evaluated in the current study (see Table 1). Note that the problems being investigated are either technical or organizational, however still in the domain of management concern.

A five-point scale was used to evaluate the current acuteness of each problem, ranging from 5 (major problem) to 1 (no problem at all). An average over all respondents provided a measure of problem acuteness. A troubling, yet not totally surprising, finding was that the state of maintenance since Lientz and Swanson's study has not changed significantly. While, on individual problem basis, there was no statistical change at the 95% confidence level in problem acuteness, there was a slight increase in problem acuteness when all problems are considered together. At higher p values of significance, problems that appear less acute now are processing and storage requirements, and inadequate user training. Maintenance programmer turnover, maintenance programmer motivation, maintenance programmer productivity, documentation quality, adequacy of system design specifications, unrealistic user expectations, and bud-

TABLE 1

An Inventory of Maintenance Problems

1. Maintenance personnel turnover
2. Documentation quality
3. System hardware and software changes
4. Demand for enhancements
5. Skills of maintenance programmers
6. Quality of original programming
7. Number of maintenance programmers availability
8. Competing demands for programmer time
9. Lack of user interest
10. System run failures
11. Lack of user understanding
12. Program storage requirements
13. Program processing time requirements
14. Maintenance programmer motivation
15. Forecasting maintenance programmer requirements
16. Maintenance programming productivity
17. Hardware and software reliability
18. Data integrity
19. Unrealistic user expectations
20. Adherence to programming standards
21. Management support
22. Adequacy of system design specifications
23. Budgetary pressures
24. Meeting scheduled commitments
25. Inadequate user training
26. Turnover in user organization

TABLE 2

The Most Serious Maintenance Problems

Problem Description	Average Rating
Demand for enhancements	3.289
Competing demands for programmer time	3.173
Documentation quality	3.173
Unrealistic user expectations	2.808
Adequacy of system design specifications	2.769
Number of maintenance programmers available	2.654
Meeting schedules commitments	2.647
Lack of user understanding	2.615
Inadequate user training	2.596
Quality of original programming	2.577

getary pressure problems appear to have become more acute over the years.

The ten most acute problems are listed in Table 2. Number one and number four on this list pertain to user demands and expectations. The MIS department is continuously flooded by demands for enhancements to the current system yet the users are very critical of lengthy, or even normal, processing delays. The users want their changes made quickly regardless of the priorities of other requests. The "responsiveness" requirements have been further accentuated with the advent of end-user computing and off-the-shelf ready-to-use software. Nevertheless, such requirements pose heavy burdens on the MIS department.

Problems ranked number two, six, and seven are: competing demands on programmer time, the availability of maintenance programmers, and meeting scheduled commitments, respectively. Maintenance has consistently demanded a significant share of IS resources (most estimates put it between 50% and 80%), yet maintenance programming is generally regarded with lack of "glamour", accorded a low priority, and not viewed as one offering much advancement potential [12,16]. New applications often take precedence over maintenance, and the best programmers are assigned or shifted to new development projects. It is justifiably arguable that this "second-class" treatment has taken its toll on the quality of maintenance.

Problems rated third, fifth, and tenth are related to the quality of the original design of the system. They include inadequate design specifications, the perennial problem of documentation quality, and the quality of original programming. Finally, many problems have to do with the lack of user understanding of and training on the information systems. Such unawareness can result in unrealistic demands and unusual requests.

PROBLEM FACTORS

A factor analysis of the responses on the twenty six problems was conducted in order to determine the underlying problem dimensions and to facilitate subsequent contingency analysis. Lientz and Swanson [9] also reported a factor analysis where they identified six underlying dimensions. However, there were potential problems in their analysis.[1]

[1]According to Lientz and Swanson, the six factors accounted for 100% of the variation. However, according to the reported eigenvalues, the six factors account for only 47% of the variation. Further, only the first three factors had eigenvalues greater than 1. Typically, factors with eigenvalues greater than 1 are the only ones retained.

TABLE 3

IS Problem Factors

Factor	Eigenvalue	Percent of Variance	Cumulative Percent
Programmer time availability	9.785	37.6	37.6
Programmer effectiveness	2.330	9.0	46.6
Operations reliability	1.662	6.4	53.0
User knowledge	1.534	5.9	58.9
Product quality	1.369	5.3	64.1
Hardware/software limitations	1.304	5.0	69.2
User/management interest	1.139	4.4	73.5

TABLE 4

The Rotated Factor Matrix

Items / Factors	Programmer Time Availability	Programmer Effectiveness	Operating Environment	User Knowledge	Product Quality	Hardware / Software Limitations	User Training
7. Number of maintenance programmers available	.8414						
8. Competing demands for programmer time	.7547						
1. Maintenance personnel turnover	.6519						
26. Turnover in user organization	.6462						
15. Forecasting maintenance programmer requirements	.6316			.4492			
14. Maintenance programmer motivation		.7840					
5. Skills of maintenance programmers		.7648					
24. Meeting scheduled commitments		.6925					
16. Maintenance programming productivity		.6832					
10. System run failures			.7356				
3. System hardware and software changes			.6979				
18. Data integrity			.6406				
17. Hardware and software reliability			.5549			.5250	
2. Documentation quality			.4411		.4212		
19. Unrealistic user expectations				.8068			
25. Inadequate user training				.7169			.4176
11. Lack of user understanding				.6670			.4176
13. Program processing time requirements				.4425			
22. Adequacy of system design specifications					.7342		
4. Demands for enhancements					.7179		
6. Quality of original programming			.4854		.6833		
20. Adherence to programming standards		.4459			.5916		
12. Program storage requirements						.7758	
23. Budgetary pressures						.7655	
9. Lack of user interest							.8710
21. Management support							.6445

Our factor analysis is exploratory as the ratio of the number of observations to the number of problem items is only 2. The general recommendation for factor analysis is to have four to five times as many observations as there are items. However, in practice, as pointed out by Hair et al. [7], several researchers have used factor analysis when this ratio is about 2. Nonetheless, with a smaller sample and a lower ratio, one has to be cautious in analysis and interpretation, as well as use lower levels of significance to increase the power of the tests [2].

Factor analysis was conducted using principal components analysis as the extraction technique and varimax as the method of rotation. Rotated factor loadings were examined to identify the constituent items of each factor. Rotated factor loadings of + .30 are considered significant, + .40 more significant, and + .50 very significant (Hair et al. 1984). In order to increase the power of the tests, we used factor loadings of .50 to denote significance.

Seven factors emerged with eigenvalues greater than one, and together explained 73.5% of the total variance (Table 3). The rotated factor matrix is shown in Table 4 (factor loadings of less than .4 are excluded for presentation clarity). The problem items are grouped under the seven factors by their highest (primary) factor loadings. Very few items had non-primary loadings, which attest to the aptness of the seven-factor model. The items that are grouped under each factor clearly point to the underlying factor domain; the factor is labeled accordingly.

The first factor is labeled **Programmer Time Availability**. The items under this factor refer to maintenance programmer shortage, their turnover, and increasing demands on them. The second factor is **Programmer Effectiveness**; it refers to programmer skills, effectiveness, and motivation. The third factor is **Operating Environment**; included are problems due to hardware and software reliability, system failures, data integrity, documentation, etc. The fourth factor is labeled as **User Knowledge**; it refers to problems caused by user expectations, lack of user training, understanding, etc. The fifth factor is **Product Quality**, and refers to the quality of the original system. It includes items such as the quality of original design specifications, quality of original programming, and adherence to standards. The last two factors are not as pure as the first five factors; however, they refer to **Hardware/Software Limitations**, and **User/Management Interest**.

For subsequent analysis, the seven problem factors were operationalized as composite indices. A factor index was computed for each survey response as a weighted sum of problem items with factor coefficients greater than .5 (see the factor matrix). These problem factors and other maintenance issues are examined on a contingency basis in the next section.

THE CONTINGENCY MODEL

The fundamental impetus behind the formulation of a contingency model is the determination of the impact of scale of effort factors such as system size, age and system complexity upon various maintenance types, problems, and issues. Three types of software maintenance are considered: **corrective, adaptive**, and **perfective** maintenance [9]. Corrective maintenance comprises of emergency program fixes and routine debugging; adaptive maintenance is making changes in response to technology changes (e.g., I/O changes, and hardware and systems software changes); and perfective maintenance includes user requested enhancements, improved documentation, and recoding for efficiency.

Systems concepts and S-curve model of technology were used to generate an initial model of contingency relationships. Specifically, systems theory can be used to make predictions relating scale of effort factors — the size, age and complexity of the system — to the various dependent variables in maintenance. The S-curve model of technology adoption life cycle can be used to relate the type of maintenance methods used to the age of the system.

From systems theory, as a system ages it becomes more disordered [4]. Negative entropy, which can be alternately referred to as maintenance, must be applied to the system or it will degrade and fail. It is, therefore, implied that an application system will need increasing amounts of corrective maintenance as the system ages.

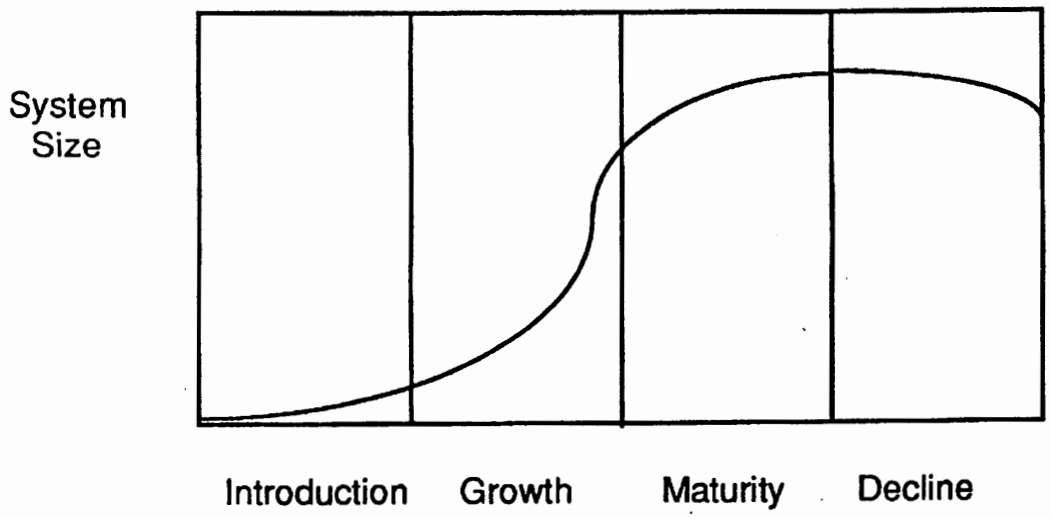
Systems which are larger or more complex have a greater chance of malfunctioning since there are more connections and interactions between system components. Large and complex systems should use more corrective maintenance [1]. Also larger systems tend to model complex, and dynamic environments. These systems will use adaptive maintenance which enables the system to adjust to the new technological environment allowing it to survive.

The S-curve is a model of organizational growth. It can be applied to show information systems (IS) growth, maturity, and decline [4,13]. In fact, Nolan's stage model of IS growth [13] can essentially be considered an application of the S-curve model. The diagram in Figure 1 describes the IS lifecycle. Each stage of the model has certain characteristic problems associated with it.

The S-curve model predicts that an IS in the maturity stage will use formal controls [13] in application development and maintenance, such as formal reviews and chargebacks. In the maturity stage, an organization becomes concerned with recovering the costs of IS; as such cost benefit analysis may more often be used as a form of maintenance control. Further, an IS in the maturity stage will employ perfective maintenance more as it attempts to obtain greater value from its existing systems.

FIGURE 1

S-curve Model of IS Growth



In the growth stage, there is a natural focus on expansion without attention to controls [13]. In this stage there will be more problems; consequently, maintenance effort will be of the corrective type. Systems in the growth stage should, therefore, be correlated with greater problems in programmer time availability, product quality, and user knowledge.

Based on the above comments from the literature, a contingency model is offered. This model (Table 5) relates organizational and system factors to the dependent maintenance factors. The independent variables are grouped into these factors: system age, system size, database size, IS staff size, type of IS, maintenance effort, and development experience. The dependent variables are: maintenance controls, maintenance problems and effort on different types of maintenance.

Many of these factors and their levels are self-explanatory. Their meanings are further explicated in Table 5. Among the independent variables, the type of information system refers to two broad categories: transaction processing/office automation, and decision support systems/executive systems/ strategic systems. Among the dependent variables, the problem types refers to the seven problem factors extracted during factor analysis. Another dependent variable is the type of controls used in maintenance. The study examined several maintenance controls. These individual controls are grouped logically into five types of controls, as listed below.

Trouble/user request logs =	{Log user requests + Trouble log + Change log}
Chargebacks =	{Chargeback equipment costs + Chargeback personnel costs}
Cost/benefit analysis =	{Cost justify user requests}
Batch maintenance implementation	
Formal audits and testing procedures =	{Formal test of changes + Formal system audit}

RELATIONSHIPS AND HYPOTHESES

Based on the contingency model and the above discussion, the following relationships and hypotheses are formulated. For space reasons, only representative hypotheses are being presented; more hypotheses can be generated by further examination of literature and experience. The discussion and the hypotheses are grouped by the independent variables.

System age

According to S-curve and system theory, system age should be correlated with the use of corrective maintenance. Older systems which are entering the growth phase of S-curve should have more problems with lack of user knowl-

TABLE 5

Maintenance Contingency Model

Independent Variables		Dependent Variables
System Age		Maintenance Controls
		Complaint / Trouble logs
System Size	—>	Chargebacks
Number of system applications		Cost Benefit Analysis
Number of application source lines		Batch Maintenance Implementation
		Formal Audit of System
Database Size		
Number of data files		Problem Types
Number of characters		Programmer Time Availability
		Programmer Effectiveness
IS Staff Size	—>	Operations Reliability
		User Knowledge
Application Development Staff Size		Product Quality
		Hardware/Software Limitations
Type of IS (TP / OA / DSS / SIS)		User Training
Relative Maintenance Effort		Type of Maintenance
		Corrective Maintenance
Development Experience of	—>	Adaptive Maintenance
Maintenance Staff		Perfective Maintenance

edge, product quality, and programmer time. The oldest systems may have problems with hardware and software limitations, but fewer problems with product quality and programmer effectiveness. Accordingly, the following hypotheses are being evaluated. (Note that, in our study, system age is measured in months and should be an indicator of the S-curve stage of the system).

H1: System Age is positively correlated with corrective maintenance.

H2: System Age is positively correlated with user knowledge problems.

H3: System Age is positively correlated with product quality problems.

H4: System Age is positively correlated with programmer time availability problems.

H5: System Age is positively correlated with hardware/software limitation problems.

H6: System Age is positively correlated with programmer effectiveness problems.

H7,8,9: System Age is positively correlated with the use of cost benefit analysis (H7), chargebacks (H8), and logging procedures (H9).

System size

System size is measured by the number of program modules and the number of source statements contained in the system. Larger systems are more complex and harder to maintain. Size should be positively correlated with corrective maintenance. Also larger systems will be more sensitive to hardware and software limitations. Accordingly, we have:

H10: System Size is positively correlated with corrective maintenance.

H11: System Size is positively correlated with product quality problems.

H12: System Size is positively correlated with user knowledge problems.

H13: System Size is positively correlated with hardware/software limitation problems.

H14,15,16: System Size is positively correlated with the use of formal maintenance control procedures like cost benefit analysis (H14), chargebacks (H15), and logging procedures (H16).

Database size

Database size is measured by the number of data files and the number of characters in the database. A larger database would seem to require greater changes in data inputs and files, as well as more frequent need for upgrades in hardware and system software. Additionally, the users are likely to be less knowledgeable/trained about the contents and use of a larger database. Thus, we have the following hypotheses:

H17: Database Size is positively correlated with adaptive maintenance.

H18: Database Size is positively correlated with user knowledge problems.

Size of staff

The number of people in the application development staff and the maintenance staff are described by this factor. While larger system staffs represent greater organizational resource allocation, they also represent larger and more complex systems. Therefore, many of the hypotheses proposed for system size should also be applicable for IS staff size. Thus we anticipate that for larger systems, there would be more problems with lack of user knowledge. However, larger staffs would be able to maintain higher product quality, and there will be fewer product quality problems. Consequently, it should result in less corrective maintenance. On the other hand, organizations with larger staff will attempt to recover or limit the costs of their staff with some formal process.

H19: Staff Size is negatively correlated with corrective maintenance.

H20: Staff Size is negatively correlated with product quality problems.

H21: Staff Size is positively correlated with user knowledge problems.

H22,23: Staff Size is positively correlated with the use formal maintenance control procedures like cost benefit analysis (H22) and chargebacks (H23).

Types of applications

Does the IS support transaction processing and office automation or does it also have features of management support systems? Examples of management support systems include decision support systems (DSS), and strategic/executive information systems (SIS). Management support systems have constantly changing requirements; as such they will generally require more perfective maintenance. Another argument for more perfective maintenance is from the point of view of the organization being in a maturity stage if it is using management support systems. Also, many times such systems require special hardware and software (e.g., workstation technology, graphical user interfaces, non-keyboard technology, etc.). As such, there are likely to be more problems with hardware and software limitations. Additionally, management support systems are primarily designed for and used by end users and managers who are generally not trained in the use of computers. Therefore, the problems of user training are likely to be heightened.

H24: SIS and DSS are positively correlated with perfective maintenance.

H25: SIS and DSS are positively correlated with user training problems.

H26: SIS and DSS are positively correlated with hardware and software limitations.

Percent of maintenance budget to overall IS budget

This factor is a measure of the relative importance of maintenance as reflected in the budget. Systems with more resources devoted to maintenance should have fewer problems.

H27: The percent of IS budget devoted to maintenance is negatively related to product quality problems.

Development Experience of the Maintenance Staff

If the maintenance staff has greater experience with system development, it is to be expected that there would be fewer problems with programmer productivity and product quality. Accordingly, we have the following hypotheses:

H28: Development experience of maintenance staff is negatively related to product quality problems.

H29: Development experience of maintenance staff is negatively related to programmer effectiveness problems.

H30: Development experience of maintenance staff is negatively related to programmer time availability problems.

CONTINGENCY RESULTS

While the hypotheses were natural to formulate based on the independent variables, it is more reflective to examine the results by each dependent variable (i.e., problem types, maintenance controls, and maintenance types). Accordingly, the hypotheses are regrouped and summarized in Table 6.

Each of the hypothesized relationship was examined by computing the Pearson's correlation coefficient. The hypotheses that are supported are included in the following discussion. While not all postulated hypotheses are supported, many are, pointing to the initial aptness of the contingency model. Future efforts should be carefully designed to further refine the contingency model.

TABLE 6

Summary of Hypothesized Correlations

(Note: All correlations are positive unless otherwise indicated.)

Problem Type Determinants

H2: User Knowledge & System Age
H12: User Knowledge & System Size
H18: User Knowledge & Database Size
H21: User Knowledge & Staff Size
H3: Product Quality & System Age
H11: Product Quality & System Size
H20: Product Quality & Staff Size (negative corr.)
H27: Product Quality & Percent Maintenance Budget (negative corr.)
H28: Product Quality & Maintenance Development Experience (negative corr.)
H5: Hardware/Software Limitations & System Age
H13: Hardware/Software Limitations & System Size
H26: Hardware/Software Limitations & DSS/SIS
H6: Programmer Effectiveness & System Age
H29: Programmer Effectiveness & Maintenance Development Experience (negative corr.)
H25: User Training & DSS/SIS

Maintenance Control Determinants

H7: Cost Benefit Analysis & System Age
H14: Cost Benefit Analysis & System Size
H22: Cost Benefit Analysis & Staff Size
H8: Chargebacks & System Age
H15: Chargebacks & System Size
H23: Chargebacks & Staff Size
H9: Logs & System Age
H16: Logs & System Size

Maintenance Type Determinants

H1: Corrective Maintenance & System Age
H10: Corrective Maintenance & System Size
H17: Adaptive Maintenance & Database Size
H19: Corrective Maintenance & Staff Size (negative corr.)
H24: Perfective Maintenance & DSS/SIS

Problem Type Determinants

System size was positively related to user knowledge problems (H12: p= .067). Other system size and age correlations were not notable. Among the individual components of the "user knowledge" factor, the "user training" component had a positive correlation with system size (p=.075) and is noteworthy. Although this was not included separately as one of the hypotheses, a plausible explanation is that large systems are unduly complex and the perceived amount of training on them is less than adequate.

Contrary to our hypothesis, the maintenance staff size and application development staff size are positively related to problems with product quality (H20: p=.03) and (H20: p=.031). The rationale for this finding may be that larger staffs work on larger and more complex systems. Larger systems are harder to maintain resulting in product quality problems. Thus, it seems that staff size is not an independent variable but an intermediate variable that is affected by system size and complexity.

The percentage of DP budget devoted to maintenance is negatively related to product quality problems (H27: p=.031). This is an expected result; a greater proportion of resources spent on maintenance should ease quality related problems. Furthermore, the development experience of maintenance personnel is negatively related to problems with programmer availability and programmer effectiveness (H30: p=.011) and (H29: p=.029). Again, these are results that confirm expectations.

The supported relationships from the above analysis are summarized in Figure 2. Several recommendations are offered in the concluding section based on these relationships.

Maintenance control determinants

Most of the relationships identified by correlation analysis support the hypotheses that were postulated. Generally, larger systems correlated with higher use of maintenance controls. For example, there was greater use of trouble/user request logs in larger systems (H14: p=.000), and also of chargebacks (H15: p=.093). There was no statistically significant support for the use of cost-benefit analysis.

Another notable result is the positive correlation between system age and the use of chargebacks (H8: p=.062). Apparently, after the system has undergone a shakeout period, users are being increasingly charged back for maintenance services.

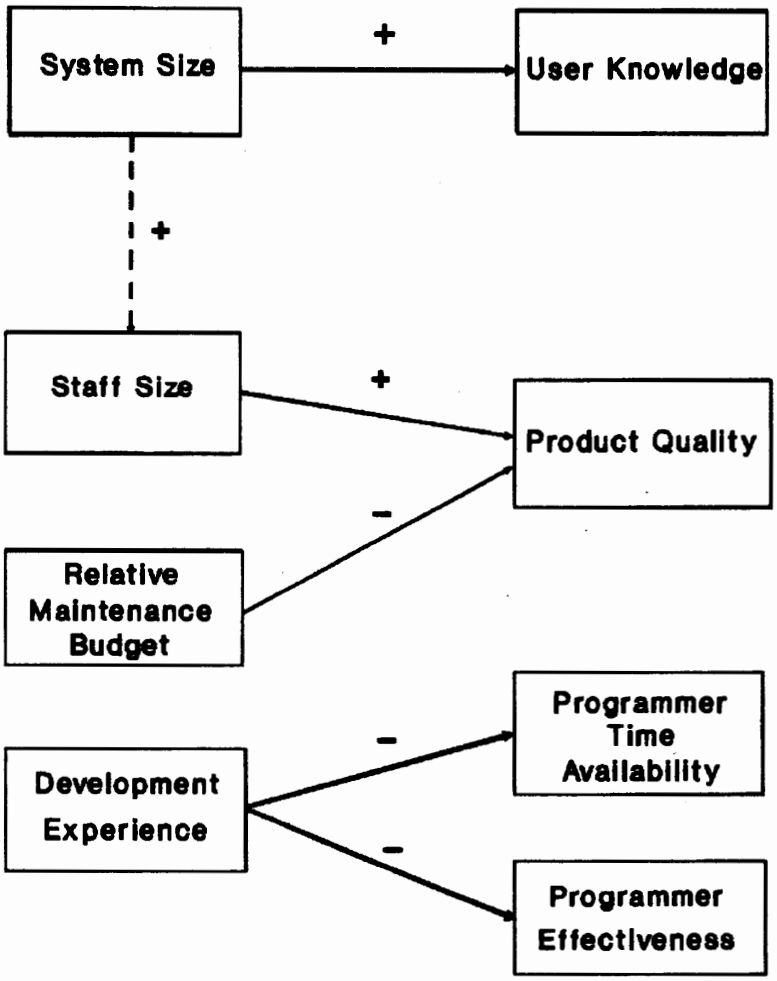
Two other interesting results that were not part of the hypotheses are: first, larger systems, as part of a control mechanism, tend to use more often the batching of maintenance changes to application programs (p = .096), and second, systems with DSS and strategic capabilities tend to use more of chargebacks (p = .002). The use of chargebacks for DSS/SIS type systems is justified as according to the S-curve model, such systems are probably in the maturity stage of IS growth in an organization, and also because such systems may not have explicit or tangible payoffs to the organization

Maintenance Effort Distribution

Once again, the findings are supportive of the hypotheses. Larger systems are positively correlated with corrective maintenance (H1: p= .088). This is in correspondence with systems theory which predicts that as a system grows larger,

FIGURE 2

Ractors Related to Problems of Maintenance



it should have more errors and corrective maintenance should increase. Systems with larger databases are positively correlated with adaptive maintenance (H17: p=.001). Larger databases may necessitate a greater amount of change in data inputs and files, as well as a greater need for hardware and software upgrades. Conversely, hardware and software changes that are made to take advantage of technological advances will impact the reorganization of larger databases more.

MANAGERIAL IMPLICATIONS AND RECOMMENDATIONS

As the study points out, despite of rapid technological advances in the software industry, problems continue to persist in application software maintenance. These problems seem to be rooted in policy and practices related to maintenance management. The following managerial recommendations, based on our analysis of the problems (see Tables 2 and 3 and Figure 2) and review of the literature, are offered as a way to alleviate the problems.

1. There seems to be a general agreement in much of the literature cited earlier that maintenance is accorded a secondary status compared to new development. It is our contention as well as of many others [9,11,12,15,16] that the lower significance attached to the maintenance function is one of the underlying reasons for the ensuing problems reported in the article. If nothing else, the simple economics of the maintenance function should force management to change this erroneous perspective. We urge a fundamental

shift in management's view of application software maintenance. Of course, this is easier said than done. Specific recommendations made below should help implement the new perspective.

2. Currently, most firms organize the maintenance function in a manner subservient to new development. That must change, and maintenance should be organized so that it has a status equal to new development. Some suggestions in this regard include: a. organize the IS function into "Installed Systems", which include corrective work as well as enhancements, and "Future Systems", which is new application development [16]; and b. take an integrated view of maintenance and new development, so that there is a seamless split (or no split at all) between development and refinement [15].

3. Refine the maintenance organization itself [12]. Key measures include: improving working conditions, offering career paths, and hiring and retaining talented and experienced employees in the maintenance department. Note that our study found that the lack of experience of maintenance staff exacerbated the problems of programmer effectiveness and availability. Other measures include providing more budget to maintenance (our study found that the relative maintenance budget had a significant relationship with product quality), instituting robust change management procedures [3], and developing measurement metrics and data [11,14].

4. Many problems related to "user knowledge" appear in the top ten list of problems, and include unrealistic user expectations, demand for enhancements, and lack of user training and understanding. Several steps can be taken to address the user knowledge category. It has been said that the maintenance group suffers from a lack of human relations management [8], and more effort should be made to achieve better communication among all concerned parties, i.e., users, managers, analysts, and programmers. A greater and effective level of communication will sensitize both users and programmers to the needs and constraints of each other. On the training issue, training is usually confined to new application systems; it should be extended to existing systems, on a periodic basis. As our study points out, training problems are more serious for larger systems; therefore, larger systems are prime candidates for user training. A partial solution to the unsatiable user demands for enhancements and new applications is user developed systems. The IS group can play a key role in improving the quality of user developed systems by providing the necessary support.

5. The maintenance activities are heavily influenced by the quality of the original design and programming [1,11]. Planning for maintenance should begin during development [5], and should include such elements as: adherence to standards, documentation, quality specifications, and sound design and programming practices based on structured principles [8].

6. New application development groups are increasingly embracing new and automated tools to boost productivity and quality; maintenance groups should do the same. Tools that support software maintenance are now available and fall into three categories: reverse engineering, software engineering, and software restructuring [3]. The use of software tools and CASE platforms that incorporate maintenance tools and software reengineering techniques has been advocated in the literature [6,12].

7. Finally, the educational system must bear some responsibility for the maintenance problems in organizations. Typically, new programmers and analysts graduating out of college, are trained only in new application development, and get only a cursory review of maintenance tasks [6]. Clearly, the MIS programs at colleges and universities need to examine their content and provide greater emphasis on the maintenance function.

CONCLUSIONS

In this article, we have provided a fresh perspective on the state of software maintenance. The previous comprehensive report was made by Lientz and Swanson [9,10] over a decade ago; essentially the same instrument that was used by them was used in this study to gather information about current maintenance issues and problems. In addition, a contingency analysis was made to explore some of the underlying reasons behind the various maintenance issues. It is disheartening to note that, in spite of technological advances, problems continue to plague maintenance activities and their management. Maintenance is fraught with many of the same problems as a decade ago, and the magnitude of the problems have in no manner diminished. The problems are exacerbated by the prevailing notion that new system development should command greater attention from management and the technical staff.

Given the vast amount of resources spent on maintenance, it is imperative from a sheer economic point of view to focus more attention on addressing the problems of maintenance. This article has highlighted the current problems in maintenance and categorized them into seven categories. Factors that contribute to the problems have also been identified. Many of the problems are non-technical in nature, and primarily require greater management attention. More than a lip service, these constantly haunting problems call for a major realignment of the management perspective on maintenance. Several practical suggestions have been offered to develop this new perspective.

REFERENCES

1. Bankar, R.D., Datar, S.M., and Kemerer, C.F. "A model to evaluate the productivity of software maintenance

projects," *Management Science*, Vol 37, No 1, January 1991, pp. 1-18.
2. Baroudi, J.J., and Orlikowski, W.J. "The problem of statistical power in MIS research," *MIS Quarterly*, 1989, Vol 13, No 1, pp. 87-106.
3. Burch, J.G., and Grupe, F.H. "Improved software maintenance management," *Information Systems Management*, Vol 10, No 1, Winter 1993, pp. 24-32.
4. Davis G. B. and Olson, M. H. (1985). *Management Information Systems* (2nd ed.). New York: McGraw-Hill.
5. Edelstein, D.V., and Mamone, S. "A standard for software maintenance: A framework for managing and executing software maintenance activities," *Computer*, Vol 25, No 6, June 1992, p. 82(2).
6. Friedlander, P., and Toothman, W.E. "Reengineering done right: Intermediate solutions that are cost effective," *Information Systems Management*, Vol 11, No 1, Winter 1994, pp. 7-15.
7. Hair, J.F. Jr., Anderson, R.E., Tatham, R.L., and Grablowsky, B.J. *Multivariate Data Analysis.* MacMillan Publishing Co. New York, 1984.
8. Kim, C., and Westin, S. "Software maintainability: Perceptions of EDP professionals," *MIS Quarterly*, June 1988, pp. 167-179.
9. Lientz, B.P. and Swanson, B.E. "Problems in application software maintenance," *Communications of the ACM*, Vol 24, No 11, November 1981, pp. 763-769.
10. Lientz, B.P., and Swanson, E.B. *Software Maintenance Management.* Addison-Wesley, Reading, MA. 1980.
11. Lientz, B.P., Swanson, B.E. and Tompkins, G.E. "Characteristics of application software maintenance," *Communications of the ACM*, Vol 21, No 6, pp. 466-471.
12. Moad, J. "Maintaining the competitive edge," *Datamation*, Vol 36, No 4, Feb 15, 1990, pp. 61-66.
13. Nolan, R. L. "Managing the crisis in data processing," *Harvard Business Review.* March-April 1979.
14. Paddock, C.E. and Shephard, G.G. "Managing software maintenance: The challenge of insufficient data," *Journal of Systems Management*, Vol 42, No 10, Oct 1991, pp. 28-31,36.
15. Poo, Danny C.C. and Layzell, P.J. "An evolutionary structural model for software maintenance," *Journal of Systems & Software*, Vol 18, No 2, May 1992, pp. 113-123.
16. Swanson, E.B. and Beath, C.M. "Reconstructing the systems development organization," *MIS Quarterly*, September 1989, pp. 293-304.

ABOUT THE AUTHORS

Prashant Palvia *is Professor of Management Information Systems at the University of Memphis. He received his Ph.D. from the University of Minnesota. He is the Editor-in-Chief of the* Journal of Global Information Management. *He was the conference chair of the 1991 International Conference of the Information Resources Management Association. His research interests include international information systems, strategic information systems, database design, and software development. He has published extensively, including in* MIS Quarterly, Decision Sciences, ACM Transactions on Database Systems, Information and Management, *and* Information Systems.

Aaron Patula *is Assistant Professor of Management Information Systems at the University of Minnesota, Duluth. His research interests are user centered design methodologies and case-based reasoning. He completed his Ph.D. in MIS with a minor in Cognitive Science from Memphis State University (now, The University of Memphis) in May 1994. The dissertation investigated case-based decision support for commodities trading.*

John Nosek *is Associate Professor of Computer and Information Sciences at Temple University. He continues to focus on the problem of obtaining greater organizational value from information technology (IT). This has led him to work on ways to align IT and organizational goals, and measuring and modifying the perceived strategic value of information systems. He has published in such journals as* Communications of the ACM, Information and Management, *and* International Journal of Man-Machine Studies.