BY THE COMPTROLLER GENERAL

Report To The Congress

OF THE UNITED STATES

Federal Agencies' Maintenance Of Computer Programs: Expensive And Undermanaged

Federal agencies spend millions of dollars annually on computer software (program) maintenance but little is done to manage it.

GAO studied 15 Federal computer sites in detail, and received completed questionnaires from hundreds of others. All reported large maintenance efforts but few had good records and very few managed software maintenance as a function.

Improvements can and should be made both in reducing maintenance on existing software and in constructing new software to reduce its eventual maintenance costs.

The National Bureau of Standards should issue a standard definition and specific technical guidelines for software maintenance. Heads of Federal agencies should require their automatic data processing managers to manage software maintenance as a discrete function.

AFMD-81-25
FEBRUARY 26, 1981

Request for copies of GAO reports should be sent to:

U.S. General Accounting Office
Document Handling and Information
Services Facility
P.O. Box 6015
Gaithersburg, Md. 20760

Telephone (202) 275-6241

The first five copies of individual reports are free of charge. Additional copies of bound audit reports are $3.25 each. Additional copies of unbound report (i.e., letter reports) and most other publications are $1.00 each. There will be a 25% discount on all orders for 100 or more copies mailed to a single address. Sales orders must be prepaid on a cash, check, or money order basis. Check should be made out to the "Superintendent of Documents".

COMPTROLLER GENERAL OF THE UNITED STATES
WASHINGTON D.C. 20548

B-201778

To the President of the Senate and the
Speaker of the House of Representatives

Computer software is the most important part of automatic data processing systems today. It is expensive to develop and maintain, and errors and omissions in software can seriously disrupt automated systems.

This report discusses the impact that computer program maintenance has on Federal computer operations, and recommends ways to improve such maintenance.

We are sending copies of this report to the Secretary of Commerce and to the Administrator of General Services.

Comptroller General
of the United States

COMPTROLLER GENERAL'S REPORT TO THE CONGRESS

FEDERAL AGENCIES' MAINTENANCE OF COMPUTER PROGRAMS: EXPENSIVE AND UNDERMANAGED

D I G E S T

Computer software maintenance consumes a large share of the Federal Government's automatic data processing (ADP) resources. After computer programs are put into operation, maintenance may be needed to make them do more or different tasks, to remove defects, or to reduce operating costs. GAO found that software maintenance has not received management attention appropriate to its cost and complexity.

GAO reviewed computer software maintenance in detail at 15 Federal computer sites and found their total annual maintenance costs to be $33 million--$19 million in programmer salaries, $8 million in other salaries, and $6 million in computer time. Two-thirds of the programmers at the 15 sites spent their time on maintenance. The Director of the General Services Administration's Software Development Office has estimated that the Government spends at least $1.3 billion annually on software maintenance. (See p. 6.)

In spite of the high cost, agencies have a very limited overview of their software maintenance operation and have made little concentrated effort to effectively manage and minimize the resources required to maintain their computer software.

Maintenance is not managed as a function. That is, ADP managers have done little either to identify common causes of maintenance problems or to take action to reduce maintenance costs. The absence of maintenance management is due in part to (a) the absence of a uniform definition of maintenance, and (b) the absence of Government-wide guidance on how to control software maintenance and reduce its costs.

Managers generally have neither cost accounting data nor management data on software maintenance activities and thus know little about how much maintenance really costs overall, or which types of maintenance cost the most. Agencies have established no goals and standards to measure the

efficiency of their maintenance operation, nor criteria for acceptable maintenance costs for given situations. They have made only limited use of improved tools and techniques which could reduce maintenance costs.

Software maintenance seems to be a common problem for all ADP users. The private sector reports that large percentages of its ADP resources are consumed by software maintenance.

Increased management attention to several problem areas could reduce costs. Inadequate emphasis in these areas appears to increase the maintenance workload either by requiring that extra maintenance be performed, or by detracting from the efficiency of maintenance that must be done.

Modifications account for about half of the total maintenance workload. While some modifications must be done to adapt software to changing user needs and prolong its useful life, others occur only because user needs were not properly identified in the first production version of the software. (See p. 17.)

Software is often maintained by people who did not develop it. If the documentation they need to understand the software is inadequate or missing, they must work harder to maintain the software. Poor documentation can increase the time to understand and maintain software applications, or lead to the redesign and rebuilding of an entire system of programs because understanding and modifying an existing program may be more trouble than building a new one. (See p. 23.)

Most data processing managers interviewed were of the opinion that contractor-developed software required more maintenance. Numerous questionnaire respondents indicated that they agreed with that opinion.

Questionnaire respondents selected better use of tools and techniques as the second most effective way to reduce maintenance. GAO has found that software tools and techniques--despite their

ability to improve the maintenance operation--are not used to their full potential at many agency data processing installations. 1/

Some organizations in the private sector have reported maintenance improvements achieved through better design and quality control in program development, increased use of tools and techniques, better documentation, and personnel-oriented measures including rotation and cross-training. (See p. 26.)

GAO developed a Provisional Checklist for Software Maintenance Management, shown in appendix I to this report. The checklist will be useful to organizations doing software maintenance.

CONCLUSIONS

Software maintenance in the Government is now largely undefined, unquantified, and undermanaged. Agencies need to develop and implement policies and procedures which will increase maintenance efficiency and ultimately reduce the amount and cost of software maintenance required. To help agencies in these efforts, standard definitions of the components of software maintenance and guidance on how to reduce its cost are needed.

RECOMMENDATIONS

GAO recommends that the Secretary of Commerce, through the National Bureau of Standards, develop and publish:

--Standard definitions of the component parts of software maintenance to aid agencies in recording and managing it.

--Specific guidance on software maintenance, detailing both how to improve maintenance of existing programs and how to construct new programs to reduce their eventual maintenance. GAO offers its provisional checklist pending action by the National Bureau of Standards.

1/"Wider Use of Better Computer Software Technology Can Improve Management Control and Reduce Costs" (FGMSD-80-38, Apr. 29, 1980).

GAO further recommends that heads of Federal agencies:

--Begin to manage software maintenance as a discrete function.

--Take measures to identify the amount of resources currently expended on software maintenance.

--Develop maintenance goals and standards as criteria to determine the efficiency and effectiveness of their software maintenance operation.

--Implement management policies and procedures to increase the efficiency of the maintenance operation and reduce future maintenance. (See app. I.)

AGENCY COMMENTS

We asked for comments from the Department of Commerce, the General Services Administration, and the parent agencies of the 15 sites at which we analyzed software maintenance in detail--listed in appendix IV. The Department of Commerce failed to furnish comments in time for inclusion in the final report. The General Services Administration, the Postal Service, and the National Aeronautics and Space Administration furnished comments in time for inclusion.

The General Services Administration (GSA) agreed with GAO's conclusions and recommendations, agreed with the definition of software maintenance, and said that they plan to assist the National Bureau of Standards in any way possible to provide guidance to Federal agencies concerning software maintenance. GSA clarified its estimate of Federal software maintenance costs by explaining that the costs do not include the software used in embedded weapons system computers.

The Postal Service agreed with GAO's overall recommendations and said that it already has measures underway in keeping with GAO's recommendations to the heads of Federal agencies. The National Aeronautics and Space Administration (NASA) expressed its concern that the definition

does not apply to its software, which is mostly used in a research and development environment. In the light of GSA's comments and NASA's concerns, GAO clarified certain details of its presentation for this final report. The changes made were not substantive. (See app. V and p. 6.)

The other agencies from whom GAO requested comments failed to respond within the 30-day period required by Public Law 96-226.

Contents

ABBREVIATIONS

ADP	automatic data processing
ANSI	American National Standards Institute
COBOL	common business oriented language
CPU	central processing unit
DOD	Department of Defense
FIPS	Federal Information Processing Standards
GAO	General Accounting Office
GSA	General Services Administration
JCL	job control language
NASA	National Aeronautics and Space Administration
NBS	National Bureau of Standards
OMB	Office of Management and Budget

CHAPTER 1

INTRODUCTION

Many Federal Government computer installations have 500 or more programs for specific applications, plus other programs acquired from the computer manufacturer for general support of computer operations. Estimates of Government spending on software range as high as $6 billion a year.

The current cumulative Federal investment in software probably exceeds $25 billion. While no exact figures are available, the General Services Administration's (GSA's) Software Development Office has estimated annual Federal software maintenance costs to be at least $1.3 billion. The software life cycle includes requirements analysis, design, development, and operation. During its operation, software requires maintenance, which includes both modification and repair. We are concerned with software maintenance because of its high cost.

THE SOFTWARE LIFE CYCLE

The complete life cycle of computer software can be divided into (1) requirements analysis, (2) system design or specification, (3) development, and (4) operation. To date, the data processing industry has emphasized the first three phases and given less attention to the operation phase, during which maintenance is done. There are several reasons for this. First, development is costly and highly visible, and the customer often demands that the developer implement the new software quickly. Second, most computer programmers consider developing new software to be more challenging and rewarding than maintaining operating software. However, much of the total life cycle cost of owning and operating software is a result of maintaining the software once it is in operation.

The Department of Defense (DOD) has recognized that software management is a major problem and has undertaken extensive efforts to address it, including issuance of DOD Directive 7920.1. This directive provides departmentwide guidance on life cycle management of automated information systems. The DOD efforts are discussed in another GAO report. 1/

SOFTWARE MAINTENANCE INCLUDES BOTH MODIFICATION AND REPAIR

For our review, we defined the application software maintenance function to include:

1/"The Worldwide Military Command and Control System--Major Changes Needed in Its Automated Data Processing Management and Direction" (LCD-80-22, Dec. 14, 1979).

--Removing defects:

(1) The software was programmed to do something other than what the user wanted.

(2) The program logic was faulty and the programs did something other than what the programmer intended.

--Tuning the software to make it more efficient and economical to operate (require less machine time and/or less computer memory to operate). 1/

--Modifying software to make it do more end-user tasks than it was originally intended to do.

--Taking miscellaneous actions such as changing the software so it will work with a new operating system.

We used this definition because (1) we believe most knowledgeable people would agree that software maintenance includes these tasks, even though there is no standard definition, and (2) we wanted to be very specific about what was included in our use of the term for data gathering purposes.

The long lives of many applications--with changes in user needs--have made software maintenance far more significant than hardware maintenance.

FEDERAL SOFTWARE MAINTENANCE IS SIGNIFICANT

Software maintenance is complex and expensive. The operations identified in our definition basically deal with (1) remedial or corrective maintenance and (2) enhancement or adaptation of the original software. To isolate and correct software defects requires a thorough knowledge of the intended use and design of the software and an understanding of the developing programmer's logic. Enhancement or adaptation requires all of the above, but is also affected by whether or not the software was originally designed to ease future changes.

To estimate how long applications programs are used (and therefore maintained), we administered a questionnaire on which we asked the average life of programs in production. (See app. II.) Respondents at 263 installations said that programs written in the

---

1/"Machine time" means those computer resources which are charged in units of time--for example, the time to actually execute instructions, which is called "central processor time" and is very costly.

COBOL programming language last an average of 5.4 years, and at 212 installations, respondents said that programs written in the FORTRAN language last an average of 4.8 years. At 399 installations respondents reported that the ages of their oldest application programs (any language) averaged about 9.4 years.

ROLES OF VARIOUS AGENCIES

The basic law governing Federal automatic data processing (ADP) management is the Brooks Act, Public Law 89-306. Under this act, the General Services Administration is responsible for coordinating the procurement and maintenance of Federal ADP resources. GSA receives technical advice from the Secretary of Commerce, primarily through the National Bureau of Standards (NBS), and both of these agencies get fiscal and policy guidance from the Office of Management and Budget (OMB). NBS is responsible for providing scientific and technological advisory services to Federal agencies and for developing Federal Information Processing Standards.

In addition, each Federal agency has certain responsibilities for managing its own ADP resources. Circular A-71, published in March 1965 by the Bureau of the Budget (later renamed the Office of Management and Budget), states that the heads of all executive departments and establishments are responsible for the administration and management of their automatic data processing activities.

In our role of aiding the Congress, we are concerned with the management of Federal ADP and with computer software as an expensive part of Federal ADP. Our past reports to the Congress have recommended improvements in ADP management both Government-wide and at specific agencies.

OBJECTIVES, SCOPE, AND METHODOLOGY

Because of the Government's large inventory of computer software and the resources necessary to maintain it, software maintenance is an area which deserves close management attention. The cost of such maintenance is substantial but opportunities exist for significant cost reductions and release of resources to other tasks. Based on this premise, we undertook this study to determine:

--To what extent are resources being devoted to application software maintenance within Federal ADP installations?

--How efficiently is the maintenance function being managed within the Government?

--What are the causes of excessive costs and problems?

--Could we suggest ways to reduce the amount of resources needed to maintain the Government's computer software?

We conducted a nationwide review which included administering a questionnaire to Federal installations, analyzing in detail the maintenance at 15 Federal installations we visited, verifying questionnaire responses at the sites visited, and studying relevant literature.

We summarized data from 409 questionnaire responses and the 15 sites we visited. Since limited resources made it infeasible to identify the total population of Federal installations doing software maintenance and to do statistical sampling with projection to the entire Government, we used a convenience sample based upon a mailing list furnished by GSA for our questionnaires. Our intent was to determine whether conditions noted at certain sites were widespread, and not to project totals for the entire Government.

Due to lack of formal management data on the maintenance process at the data processing installations we visited, we had to develop much of the information at source levels. We did this by interviewing data processing personnel at the working, working supervisor, and management levels. Percentage of resources devoted to maintenance, where not furnished by the agency, was calculated as a weighted average from input from each organization having maintenance responsibility. Maintenance costs, where not furnished by the agency, were calculated by applying the percentage of resources devoted to maintenance to the total actual, or average, costs furnished by the agency for a particular cost category.

We also examined current literature to (1) compare the Federal level of maintenance effort to others reported, (2) provide background for suggestions for improvement, and (3) assemble a modest bibliography on the subject which would be useful to others.

CHAPTER 2

SOFTWARE MAINTENANCE: A HIGH-COST AREA

Although no one knows exactly what the Federal Government spends for software maintenance, we believe it costs a great deal and diverts a large fraction of Federal ADP resources from more productive work. We developed our own estimates at 15 sites: the total annual maintenance costs were over $33 million. The Director of GSA's Software Development Office told us he has estimated the Government's annual software maintenance costs to be at least $1.3 billion. These high dollar costs are further supported by the resources consumption reported on questionnaires from 409 sites--over half their programmers' time is devoted to maintenance. This situation is not unique to the Federal Government. Private sector sources also report in the literature that half or more of their programmers' time is devoted to maintenance.

DOLLAR COST OF SOFTWARE MAINTENANCE IS HIGH AT THE SITES WE VISITED

After developing percentages of resources consumed, we applied them to actual or average salaries and hourly computer charges to estimate annual costs of personnel and hardware categories.

Category	Amount
Personnel:	
Programmer/Analyst	$19,385,873
Operations	1,356,887
Support (administrative)	1,549,067
Management	1,111,378
Combined (note a)	3,577,556
Total personnel	$26,980,761
Hardware	$ 6,320,913

a/Sites reporting maintenance personnel cost in total only.

The size of these annual expenditures, which represent only 15 out of the many Federal data processing installations, together with the high percentages of resource consumption reported on our 409 questionnaires from Federal installations, emphasizes that software maintenance is a significant cost area in Federal ADP installations and must be managed effectively. Local management needs more detailed information on which to base actions to reduce maintenance costs.

GSA'S GOVERNMENT-WIDE ESTIMATE IS $1.3 BILLION PER YEAR

The Director of the Software Development Office told us that, as a rough estimate, software maintenance costs the Federal Government at least $1.3 billion per year. He explained that there are no exact figures available, but conservative estimating techniques were used to arrive at that minimum figure.

He also explained that the maintenance in that figure is made up of two major parts--one, work done to adapt software to changing user needs and extend its useful life; the other, work done to fix software errors and other problems.

At a later meeting, he said that his $1.3 billion figure does not include maintenance of software used in computers that are embedded in weapons systems. The Department of Defense is making large-scale efforts at software improvement, including developing a new programming language. These efforts were mentioned in our software technology report 1/ and discussed at length in another GAO report. 2/

HIGH PERCENTAGE OF DATA PROCESSING RESOURCES IS DEVOTED TO SOFTWARE MAINTENANCE

At the 15 sites we visited, about two-thirds of the programmer and analyst labor is devoted to maintenance--about $19.5 million out of their total maintenance direct labor cost of about $27 million per year. We had to develop this information ourselves, and did so in the following way.

To estimate how significant software maintenance is in Federal installations, we first needed to identify the percentages of data processing resources devoted to maintenance. The following percentages of personnel and hardware resources were taken from our questionnaire responses and determined by us at the data processing installations we visited.

1/"Wider Use of Better Computer Software Technology Can Improve Management Control and Reduce Costs" (FGMSD-80-38, Apr. 29, 1980).

2/"The Department of Defense's Standardization Program for Military Computers--A More Unified Effort is Needed" (LCD-80-69, June 18, 1980).

Category	Percentage of time spent on maintenance: Average of 352 questionnaires	Average of 15 sites reviewed
Personnel:		
Programmer/Analyst	52.9	66.1
Operations	8.9	11.7
Support (administrative)	5.8	33.9
Management	10.0	26.9
Hardware	13.6	13.6

We believe the reason the percentages for the installations we visited are generally higher than those from the questionnaires is that the agencies answering our questionnaire may have omitted some activities from their definition of software maintenance. The most likely omission is user-requested modifications to existing software, which some agencies consider development work. At the sites we visited, we were able to assure ourselves that the data used to calculate the percentage of resources were based on all the activities in our definition of software maintenance. Despite their somewhat lower reported percentages, we feel that the questionnaires agree substantially with the sites visited.

As software is a labor intensive activity, it was predictable that the largest block of resources would be in the personnel area, particularly in the programmer/analyst category. At the 15 data processing installations visited, about 66 percent of the programmer/analyst time was required for maintenance. The average percentage reported by the 353 installations who answered this question on our questionnaire was about 53 percent. Additional percentages were stated for various other categories of personnel.

The percentage of computer time devoted to testing maintenance changes to programs was about 13 percent according to our questionnaire respondents, and also averaged about 13 percent at the installations we reviewed. This amount of computer time not only represents a high dollar cost (see p. 6) but also diverts computer time that could be used for other purposes.

The resources listed do not include such items as facility overhead, since generally not enough cost data were available to accurately allocate these resources to software maintenance. From the figures obtained, however, it is evident that a large segment of the Government's data processing resources is used to maintain its computer software.

PRIVATE SECTOR ALSO DEVOTES LARGE RESOURCES TO SOFTWARE MAINTENANCE

Computer software literature has traditionally focused on software development but now shows a growing concern about the

increasing software maintenance burden in the private sector. Recently, recognition of software maintenance as a major contributor to increasing operating cost, and the exploration of ways to reduce it have become the subjects of frequent articles in ADP publications. Published estimates of programmer time spent on the maintenance function range from 50 to 80 percent. Perhaps even more significant, some estimates say that up to 60 percent of all ADP dollars will be spent on software maintenance in the future if the present growth rate continues. Some reported levels of resources used for software maintenance are shown below.

A survey which reported application software maintenance in 69 organizations 1/ found that annual labor hours were allocated thus:

Activity	Percentage
Maintenance and enhancement	48.0
New development	46.1
Other	5.9

The same survey subdivided software maintenance thus:

Category	Activity	Relative frequency (percentage)
Corrective	Emergency fixes, routine debugging	17.4
Adaptive	Accommodation of changes to data and files, and to hardware and system software	18.2
Perfective	User enhancement, improved documentation, recoding for computational efficiency	60.3
Other		4.1

Reports by other sources on maintenance effort levels include the following:

--Modifying existing application systems to incorporate such changes typically requires about one-half of an organization's programming effort. Many companies have adopted the

---

1/See app. III, ref. 25.

practice of setting a budget for maintenance activities and then doing just the highest priority jobs that can be accomplished within this budget. 1/

--One recent DOD study showed that the cost of development for Air Force avionics software averaged about $75 per instruction while the cost of maintenance corrections of deployed software has ranged up to $4,000 per instruction. 2/

1/See app. III, ref. 36.

2/See app. III, ref. 21.

CHAPTER 3

NEED FOR INCREASED MANAGEMENT ATTENTION TO SOFTWARE MAINTENANCE

Our study shows that software maintenance is not managed as a function at many Federal installations and that excessive costs can result. Many Federal ADP managers have little data on their maintenance costs or which types of maintenance are most common and have no standards or goals for software maintenance. The failure to manage software maintenance stems at least in part from the lack of a standard Government-wide definition of the components of maintenance and the lack of central guidance on how to manage software maintenance and reduce its costs.

If software maintenance were managed as a function, its cost could be reduced in several ways: (1) controlling excessive user-requested modifications, (2) more accurately defining user requirements during system development, (3) requiring better documentation, (4) exerting better controls on contractor software developments, and (5) making better use of software tools and techniques. To provide interim assistance, we developed a Provisional Checklist for Software Maintenance Management (app. I), which summarizes ways to control and reduce software maintenance costs.

SOFTWARE MAINTENANCE IS NOT MANAGED AS A FUNCTION

We found that Government managers were not managing software maintenance as a function. That is, maintenance processes were not identified, grouped, documented, and reported so that management could have a comprehensive picture of the installation's total software maintenance efforts. Without such a picture, it is impossible to measure performance. If performance cannot be measured, poor utilization of resources can go undetected for long periods, resulting in failure to meet objectives. Also, management has insufficient information to help it decide how to correct perceived deficiencies. This absence of definitions, cost records, and goals or standards makes it virtually impossible to manage software maintenance as a function.

In our guidelines for accounting for ADP costs, 1/ we cited the necessity for segregating activities into "work functions" as a prerequisite for effective management of ADP costs. Accumulating activities by work function permits an evaluation of the

---

1/"Guidelines For Accounting For Automatic Data Processing Costs" (Federal Government Accounting Pamphlet No. 4, 1978).

efficiency of performing specific operations and a comparison of the cost of functions that can be accomplished in more than one way. We believe that software maintenance should be identified and managed as a discrete function because of its high cost.

DEFINITION AND GUIDANCE ARE LACKING

The absence of maintenance management is due at least partly to the absence of definition and guidelines.

Maintenance is not uniformly defined

The definition and concept of software maintenance vary from agency to agency. In some cases, we found inconsistent definitions of software maintenance within the same agency. To insure uniform data for this review, we requested that all data provided reflect the operations included in our software maintenance definition, regardless of the agency's own definition. While a number of data processing managers agreed that our definition was generally accepted in the data processing industry, only one of the 15 installations we visited had formally defined maintenance in this manner. We found the following variations in the definition of what did--or did not--constitute maintenance at various agency sites:

--Modifications to existing software are considered development instead of maintenance.

--Work done on software applications developed centrally, but run at the installation, is not considered maintenance.

--Only defect removal is considered maintenance.

--Maintenance is not formally defined at all.

There are adverse effects at both installation and Government-wide levels from not having software maintenance properly defined, managed, and understood. At the installation level, policies and procedures for the utilization of maintenance resources may be based on incomplete, inconsistent, or erroneous data. When individuals doing maintenance work report their work based on only a partial definition or on no definition of maintenance, each individual may report just those activities which agree with his own concept of what maintenance includes. When this occurs a data processing manager does not have the correct operating picture for his installation and cannot make informed management decisions.

On a Government-wide level, variances in the definition of maintenance make it difficult for central Government agencies to develop statistics and analyze the maintenance function in the Government, and to issue across-the-board guidance, since the terminology they would use would not mean the same thing to all agencies.

We consulted publications of the National Bureau of Standards and the American National Standards Institute (ANSI), seeking definitions of software maintenance. While NBS publications 1/ mention an "operation and maintenance" phase of the software life cycle and an ANSI publication 2/ has a definition of "maintenance"--which seems mostly oriented to repair of hardware--we found no explicit definition of software maintenance published by either source.

We met informally with officials of the NBS Institute for Computer Science and Technology in May 1980 and June 1980, to discuss software maintenance and what we think is needed. The representatives we spoke to did not have current publications with an explicit definition and did not show us any projects which explicitly address software maintenance as a discrete function. However, they did show us a draft of a conversion document, expected to be published in July 1980, which included a brief definition of software maintenance to distinguish it from conversion. 3/

We feel that brief definition must be amplified to define maintenance in its own right, especially for cost accounting purposes. The Institute representatives indicated they felt that a standard definition for Government-wide use should consist of a list of software maintenance components from which installations could select those relevant to them.

Central agency guidance is lacking

While some Government publications have implications for software maintenance--such as the Federal Information Processing Standards Publications (FIPS PUBs) on documentation--we have found no documents from NBS or GSA specifically devoted to software maintenance. We believe that such a costly area deserves separate, explicit treatment as a subject in its own right.

At our meeting with NBS officials they said, and we agree, that software maintenance is implied in some of their publications and that some of their projects--such as their publication on verification, validation, and testing 4/--would have a beneficial effect on software maintenance by eventually yielding better software development.

1/See app. III, refs. 7 and 8.

2/ANSI dictionary (adopted as FIPS PUB 11-1).

3/Since then, it has been published as "Conversion of Federal ADP Systems: A Tutorial" (NBS Special Publication 500-62).

4/"Verification, Validation and Testing for the Individual Programmer" (NBS Special Publication 500-56).

We believe that the absence of guidance specifically devoted to software maintenance contributes to agencies' failures to keep track of it, set goals for it, and manage it as a function.

MANAGEMENT DATA AND STANDARDS ARE LACKING

The absence of maintenance management is shown by the common lack of data on costs and types of maintenance being done and the common absence of standards or goals for software maintenance performance.

Little management data on costs of maintenance or types of maintenance being done

At the 15 data processing installations reviewed, we found no system to accumulate and report management data that would identify the extent of resources devoted to the software maintenance function.

Managers cannot cost and size their overall maintenance effort or evaluate its efficiency without this data. Attempts to reduce maintenance may fail because of lack of information.

None of the 15 data processing installations we visited had cost accounting systems in place to capture and report software maintenance costs. Some installations accounted for personnel time spent directly on general maintenance functions, but did not account for hardware or indirect costs associated with maintenance. The other installations visited made no attempt to capture any maintenance costs. (One-third of the respondents to our questionnaire indicated they had a cost system that considered software maintenance costs.)

Costs associated with the software maintenance function should be monitored and recorded. Segregating the costs of the different work functions involved in data processing is a prerequisite for effective management of ADP costs. Some specific reasons for accumulating such data are:

--Knowledge of costs is necessary to estimate the feasibility of requests for maintenance work.

--Cost accounting would enable management to identify all work segments which contribute to maintenance costs.

--Accounting for maintenance costs would permit evaluation of the efficiency of specific operations, and comparison of the costs of functions that can be accomplished in more than one way.

--Cost information is necessary in reporting and billing costs to users.

--Maintenance cost information could provide a basis on which to evaluate individual performance of personnel involved in software maintenance.

--Costs of maintaining individual user applications can alert managers to high cost and demand areas in their software inventory warranting attention.

We found that data processing managers have little knowledge of which types of maintenance cost the most in their operation. In some cases, general distinctions could be made between modification maintenance and repair maintenance, but no formal tracking systems exist to provide the degree of detail necessary for meaningful management analysis of the maintenance workload. The significance of this becomes apparent when it is considered in terms of management goals to increase efficiency or to reduce the maintenance resources needed.

In "Guidelines For Accounting For Automatic Data Processing Costs" 1/ we recommended that distinct records be kept on what is spent on software maintenance.

To use his resources more efficiently, a manager must first know where they are going. Second, to reduce the number of maintenance actions being performed, the causes of those actions must be identified and dealt with. Each type of maintenance action identified may have some preventable causes. Without knowing the types and quantities of software maintenance in his total maintenance workload, a manager cannot identify high cost areas and thus cannot act to reduce them. Also, individual programmer performance is seldom if ever measured in such areas as time spent to perform maintenance tasks.

To better define the software maintenance workload we constructed a profile of the maintenance activity at each data processing installation visited. We also asked that each questionnaire respondent estimate the percentage of maintenance workload spent on each activity.

1/App. III, ref. 4.

Category	Average percentage spent on each maintenance activity: Installations visited	Questionnaires
Defect removal	20.3	18.9
Tuning	7.4	11.1
Modification	61.3	50.7
Other	9.3	19.2

The mixture of maintenance activities varies from installation to installation. For this reason, an installation manager must analyze his own maintenance workload mixture to determine the types of maintenance which consume most of his resources.

Some general trends can be seen from the above data. Modifications--meaning changes to what the software does for the user--appear to constitute most of the maintenance workload, while the other activities are less important overall, even though they may appear in different proportions at individual sites.

No agency maintenance goals and standards in use

None of the 15 agency data processing installations where site work was conducted had established goals or standards specifying acceptable levels for software maintenance activity. Such standards would give agency managers a basis on which to determine whether their present levels of maintenance are efficient and effective. Standards would reflect the percentage of the installation's resources which could reasonably be required to maintain the installation's software applications, and would be based on a thorough analysis of current workload and carefully documented historical maintenance data covering representative past periods. In establishing acceptable levels for maintenance expenditures at a particular installation, those factors unique to that installation must be identified and considered. Examples of such factors are

--programming languages used,

--age of software applications,

--program complexity,

--frequency of user requirement changes, and

--quality of documentation.

Without benefit of such standards, an installation manager has little or no basis for evaluating the efficiency and effectiveness of his present level of maintenance.

The lack of definitions and accounting data contribute to the lack of goals. Without information management cannot set meaningful goals.

Agency goals should include reducing the level of resources necessary to maintain a given software inventory. The reduction in resources would be made possible by planned management actions to increase efficiency and minimize the amount of maintenance required.

PRIVATE SECTOR IDENTIFIES SOME COMMON CAUSES OF HIGH SOFTWARE MAINTENANCE COSTS

In the survey quoted in chapter 2, 1/ the 69 organizations showed the following as the most important factors in software maintenance costs.

Rank	Cause
1	User demands for enhancements
2	Quality of system documentation
3	Competing demands on maintenance personnel time
4	Quality of original programs
5	Meeting schedule commitments
6	Lack of user understanding of system

Turnover of maintenance personnel and maintenance programming productivity, which are often advanced as reasons why maintenance is expensive, ranked lower: 9th and 20th, respectively.

Other literature agreed with the above survey that user demands are most important and that documentation and quality of original programs are important, but attached more importance to programmer motivation.

SOFTWARE MAINTENANCE COSTS COULD BE REDUCED

Increased management attention to several problem areas identified at the data processing installations reviewed could reduce costs. Inadequate emphasis in these areas appears to increase the maintenance workload either by requiring extra maintenance or by detracting from the efficiency of the maintenance that must be done.

1/See app. III, ref. 25.

Excessive user-requested modifications

Some formal review and approval procedures are usually in place for maintenance efforts which require large amounts of resources. Smaller tasks, however, tend to be handled informally with little management review and approval required. Where no approval is required, or where management approval is "rubber stamped," users may submit unlimited requests for maintenance. As an example, at one location we found management approval for maintenance requests was based on whether time was available to do the work, and not on the need for the change. We noted that a single application at this installation had 158 modifications documented in its maintenance history.

Inadequate definition of user requirements in system development phase

We found that modifications account for about half of the total maintenance workload. While some modifications to software applications must be done to adapt them to changing user needs and prolong the useful life of the software, others occur only because user needs were not properly identified and addressed in the first production version of the software. Of our 409 questionnaire respondents, 171 indicated that better definition of user requirements in the system development phase would be the single most beneficial type of effort to reduce software maintenance.

Inadequate or missing documentation

The ADP term "documentation" refers to the information recorded during design, development, operation, and maintenance of computer software to explain pertinent aspects such as the purposes, methods, logic relationships, capabilities, and limitations of the software.

Software is often maintained by people who did not develop it. If the documentation they need to understand the software is inadequate or missing, they must work harder to maintain it. Results of poor documentation have ranged from increased time to understand and maintain software applications, to complete redesign and rebuilding of an entire system of computer programs because understanding and modifying the existing one was more trouble than building a new one.

At the data processing installations visited, several managers conceded that poor documentation adds to software maintenance costs at their installation. Besides a lack of program development documentation, we found a lack of secondary documentation (such as maintenance histories) on individual software applications.

Inadequate control of contractor software developments

At installations where both contractor-developed and inhouse-developed software is run, most data processing managers were of the opinion that contractor-developed software required more maintenance. Questionnaire respondents also indicated that contractor-developed software usually requires more maintenance. There can be several reasons for this. The two basic reasons, however, are often that the agency fails to insure that the contractor has a good quality assurance program in effect during development, and that agency personnel who must later maintain the software not only learn nothing about it while the contractor is developing it but also inherit little or no documentation.

Our software contracting report 1/ discussed several cases where software contracts delivered unusable software.

Limited use of software tools and techniques in the development and maintenance of software

Until recently, software development was considered an art by management and left to the control of technicians. Software which cost too much to develop, operate, and maintain was one result of this practice. Improved software tools and techniques, which can aid in the development and maintenance of computer software, have been developed in an effort to better manage software.

Questionnaire respondents selected better use of tools and techniques as the second most effective way to reduce maintenance. Despite their potential for improvement in the maintenance operation, we have found software tools and techniques are not used to their full potential at many agency data processing installations. 2/

A software tool is a computer program that can automate some of the labor involved in the management, design, coding, testing, inspection, or maintenance of other programs. A wide range of these tools is now available commercially. Some common tools are:

--Preprocessors. Preprocessors perform some preliminary work on a draft computer program before it is completely tested on the computer. Types of preprocessors include "filters" (also known as code auditors) which allow management to determine quickly whether programmers are obeying

---

1/"Contracting For Computer Software Development--Serious Problems Require Management Attention To Avoid Wasting Additional Millions" (FGMSD-80-4, Nov. 9, 1979).

2/FGMSD-80-38, op. cit. (See p. 6.)

specifications and standards, and shorthand preprocessors which allow the programmers to write the programs in a very abbreviated form which is then expanded by the preprocessor before it is tested on the computer. Shorthand preprocessors reduce writing, keypunching, and proofreading labor.

--Program analyzers. These tools modify, or monitor the operation of, an applications program to allow some information about its operating characteristics to be collected automatically. This information can then be used to help modify the program to make it cost less to run on the computer, or to verify that the program operates correctly.

--Programmer support libraries. These are automated filing systems which can support the programming development projects of entire installations. A programmer support library maintains files of draft programs, data, and documentation, and can be used to provide management with progress reports.

--On-line programming support programs. These tools allow programmers to quickly correct and modify application programs and quickly test program results.

--Test data generators. These analyze a program and produce files of data needed to test the logic of the program.

Software techniques are methods or procedures for designing, developing, documenting, and maintaining computer programs or for managing these activities. There are generally two types of software techniques: those that are useful to, and done by, persons who work on programs, and those that are useful to managers to control their work. Examples of software techniques useful to workers include:

--Structured programming (also called structured coding). A technique of developing computer programs so that they will be more easily understood by others who must later maintain and modify them. Such easier understanding aids documentation, testing, and correction.

--Top-down program development. Designing, coding, and testing systems by building program modules starting with those at the general level (the "top") and proceeding down to the most specialized, detailed level (the "bottom").

--Performance improvement. Analysis and subsequent modification of computer programs to make them cost less to run on a computer while still giving the same user answers. Performance improvement may be aided by various software tools, including program analyzers (see above).

--Concurrent documentation. Developing documentation at the same time as the program is being developed to provide better project control, aid completeness of the documentation, and save money.

Examples of techniques useful to management include:

--Requiring independent inspection of software by someone other than the developer. This improves software quality by imposing discipline on the developer. It is now feasible to require such inspection because current tools can automate much of the work involved.

--Using the chief programmer team method of organizing programming projects. The team nucleus includes a very skilled chief programmer, a backup programmer, and a programming librarian.

--Making a deliberate effort to find, analyze, and, if suitable, use existing software instead of developing new software for the same purpose. This applies both to software tools and to applications software.

Maintenance-related benefits from the proper use of software tools and techniques are:

--Data processing managers can substantially improve control over the maintenance operation.

--Overall maintenance costs may be reduced. We recently reported savings at one installation of an estimated $5 million in maintenance costs in 4 years using a combination of software tools and techniques.

--Structured programming can both reduce errors and make programs easier to modify.

--Appropriate tools can automatically provide information for use in the modification of programs to make them less costly to run on the computer.

A particular area in which tools and techniques could aid maintenance is in the testing of maintenance changes before they are put into production. We found that little such testing is done.

PRIVATE SECTOR TRIES SEVERAL MEASURES TO IMPROVE SOFTWARE MAINTENANCE

Successful approaches to reducing software maintenance costs and problems reported in the publications we reviewed included:

--Better design and quality control in program development, including designing software to be maintainable and using quality control groups that are independent of software developers.

--Use of structured programming and higher level languages. (Structured programming can reduce the complexity of software, making modification easier. Higher level languages can improve portability and maintainability by removing concern with machine properties and features which are not relevant to the application.)

--Better documentation throughout the life of the software.

--Use of software tools and techniques to make thorough inspection and testing more feasible during development, to make more thorough testing of maintenance changes convenient, and to reduce labor generally in both production and maintenance.

--Personnel-oriented approaches, including (1) involving the productive maintainers in acceptance testing and inspection of new software, (2) rotating programmers between development and maintenance, and (3) requiring more than one programmer to maintain a given system to reduce the number of programs that can be understood by only one person.

OUR PROVISIONAL CHECKLIST SUMMARIZES SOME USEFUL MEASURES

For use until specific guidance on controlling and reducing software maintenance can be issued by NBS, we prepared a Provisional Checklist for Software Maintenance Management (app. I to this report) which lists matters we feel agency ADP management should consider to control and reduce their software maintenance workload. To achieve this objective requires three basic actions--(1) identifying significant software maintenance activities, (2) associating those activities with causes or reasons why they must be done, and (3) acting to reduce or eliminate those causes where possible.

Our checklist is in four parts--(1) recording the maintenance workload, (2) analyzing the maintenance workload, (3) increasing efficiency in the present maintenance operation, and (4) reducing future maintenance. Recording and analyzing the maintenance done now are necessary both to improve or reduce present maintenance and identify actions which can be taken during development of new software to reduce its eventual maintenance costs. Because software development has strong influence on software maintenance, we discuss what may be done during development to reduce eventual maintenance. The actions during development may be taken with either in-house or contractor-developed software.

While our checklist is only an interim document, we feel that it will be useful to persons involved with computer software maintenance. The levels of effort and emphasis devoted to specific items mentioned will vary with the type and size of specific applications.

CHAPTER 4

CONCLUSIONS, RECOMMENDATIONS, AND AGENCY COMMENTS

CONCLUSIONS

Software maintenance is a high-cost area which has not been receiving adequate management attention. Improvement is needed in the development and implementation of policies and procedures which will provide management with key data on maintenance, increase maintenance efficiency and effectiveness, and ultimately reduce the amount of software maintenance required.

The maintenance effort in the Government is now largely undefined, unidentified, unquantified, and undermanaged. Performance evaluation of the maintenance function by Government data processing managers is almost nonexistent. Data collection mechanisms are not in place to gather and report complete management data on the various aspects of application software maintenance. Consequently, data processing managers do not have sufficient information to determine the exact types of maintenance being performed, or the costs of maintenance. Without such information, managers cannot identify preventable causes of maintenance, establish criteria for acceptable levels of maintenance work, or discern remedial actions which will reduce maintenance costs.

Although our work was done at the data processing installation level, we recognize that broad and comprehensive policies which will ultimately reduce the amount of resources needed for software maintenance must come from higher levels of agency management. The data needed to formulate those policies, however, must be obtained at the installation level where maintenance is being done. In order to make informed decisions to reduce software maintenance resources, management must know

--what resources are being expended on software maintenance and the cost of those resources,

--what is the efficiency of the maintenance operation,

--what type of maintenance actions make up the total maintenance workload, and

--what are the earliest preventable causes which required the maintenance actions to be performed.

Once this management data is available, procedures and policies can be implemented which will increase the efficiency of mandatory software maintenance and also eliminate the necessity for much of the maintenance now being done. The net effect of these management actions would be to control and eventually reduce the amount of data processing resources currently being expended on software maintenance by the Federal Government.

RECOMMENDATIONS

We recommend that the Secretary of Commerce, through the National Bureau of Standards, develop and publish:

--A standard definition of applications software maintenance for Government-wide use. The publication should list and define maintenance components suitable for use in recording costs, from which individual installations can use the parts that are relevant to them.

--Guidance specifically and explicitly directed at techniques for reducing Federal software maintenance costs. Pending such publication, we feel that our provisional checklist (app. I) will be useful to installation managers who want to reduce their maintenance costs.

We further recommend that heads of Federal agencies:

--Begin to manage software maintenance as a discrete function; that is, to consider maintenance as a high-cost area needing comprehensive management policies that deal specifically with its issues. To accomplish this, data gathering mechanisms must be put in place to provide management with information on the maintenance workload.

--Identify and assign costs to resources expended for software maintenance. A suggested methodology to record costs is outlined in appendix I. Accounting and reporting of costs by area of management responsibility are fundamental steps in making individuals conscious of and responsible for the costs incurred within their area of control.

--Develop maintenance standards and goals as a means of evaluating maintenance efficiency and for use as a management tool. After carefully analyzing the current maintenance workload (see app. I), management should set goals reflecting the resource usage considered reasonable to maintain the current inventory of software. Levels of resources above these standards would be subject to management attention and subsequent action. Maintenance goals should reflect a lower level of resources expected to be attained by increased efficiency and by the use of techniques to reduce the need for future maintenance.

--Implement policies and procedures to increase the efficiency of the software maintenance operation and reduce the amount of software maintenance needed in the future. Suggested methods are listed in our Provisional Checklist for Software Maintenance Management (app. I).

AGENCY COMMENTS AND OUR EVALUATION

We asked for comments from the Department of Commerce, the General Services Administration, and the parent agencies of the 15 sites at which we analyzed software maintenance in detail--listed in appendix IV. The General Services Administration, the Postal Service, and the National Aeronautics and Space Administration furnished comments in time for inclusion. Their replies are included as appendix V of this report and discussed below. The other agencies from whom we requested comments failed to respond within the 30-day period required by Public Law 96-226.

General Services Administration

The Administrator of General Services agreed with the report's findings and stated that his Office of Software Development will assist the National Bureau of Standards and other Federal agencies in this area.

Concerning our finding that software maintenance is a high-cost area which has not been receiving adequate management attention, the Administrator said that he agreed and that GSA believes software maintenance should be considered as a unique element in the context of the overall software management problem.

Concerning our definition of software maintenance, the Administrator said he believes that the data processing community understands the term as we defined it. He said the fact that software maintenance includes elements akin to software development and conversion illustrates that software activities must be managed as an entity.

The Administrator went on to say that it is in response to the previous lack of attention to the software area that GSA recently established the Office of Software Development to provide a specialized center of software expertise for agency assistance, that software maintenance is one element which that Office's planned activities will address, and that GSA plans to assist NBS in any way possible to provide guidance to Federal agencies concerning software maintenance. We believe that the Office of Software Development will be able to provide valuable services both in the publication of guidance and in assistance to individual agencies.

The Administrator concluded his written comments by suggesting three points which he believed we should emphasize. We agree and have done so where appropriate in appendix I to this report.

U.S. Postal Service

The Postmaster General said the Postal Service agreed with the report's overall recommendations and already has measures

underway in keeping with our recommendations to heads of agencies. Specifically:

1. The Postal Service has centralized its control for maintenance requests for national systems, and is installing detailed procedures for maintenance projects of all sizes, for managing software maintenance as a discrete function.

2. The new maintenance management procedures will address identification of resources spent on software maintenance.

3. The Postal Service is now measuring and analyzing its maintenance workload and will have developed standards and goals by early 1981.

4. The new maintenance management is expected to increase the efficiency of the maintenance operation and reduce future need for maintenance.

National Aeronautics and Space Administration

The National Aeronautics and Space Administration (NASA) commented that it is in favor of voluntary guidelines with a life cycle perspective and expressed concern about (1) our definition of software maintenance, (2) the differences between research and development software and administrative software, and (3) isolating the software maintenance function from the overall life cycle management function.

We believe the fact that NASA disagrees while GSA agrees (p. 25) illustrates exactly our point that there is no standard definition and that work should be done to define the components of software maintenance. We are aware that some research and development software goes through many modifications and feel that this area should be specifically addressed in published guidance on software maintenance. We are also aware of "routine" tasks such as statistical calculations within the research and development area and believe that our maintenance management suggestions could be applied to them. Furthermore, NASA no doubt has business applications--such as payroll--to which, even in NASA's own perception, our maintenance management suggestions could be applied.

Concerning discrete management--the "isolating" spoken of in NASA's third concern above--we believe that an activity which many people agree is half or more of the total life cycle cost of software deserves explicit management as a subject in its own right. We do not, however, propose that it be isolated from the life cycle context; indeed, maintenance considerations are woven into the entire life cycle--for example, actions taken, or not taken, during the development phase will affect future maintenance during the production phase.

APPENDIX I

PROVISIONAL CHECKLIST FOR

SOFTWARE MAINTENANCE MANAGEMENT

RECORDING THE MAINTENANCE WORKLOAD

ANALYZING THE MAINTENANCE WORKLOAD

INCREASING EFFICIENCY IN THE PRESENT MAINTENANCE OPERATION

REDUCING FUTURE MAINTENANCE

PROVISIONAL CHECKLIST FOR

SOFTWARE MAINTENANCE MANAGEMENT

This checklist, which we prepared during our review, lists matters which we feel agency ADP management should consider to reduce their software maintenance workload. To achieve this objective requires three basic actions--(1) identifying those activities which are significant in the current local maintenance workload, (2) associating those activities with causes or reasons why they must be done, and (3) acting to reduce or eliminate those causes.

This checklist is in four parts--(1) recording the maintenance workload, (2) analyzing the maintenance workload, (3) increasing efficiency in the present maintenance operation, and (4) reducing future maintenance. Recording and analyzing the maintenance done now are necessary both to improve or reduce present maintenance and to identify areas where action can be taken during development of new software to reduce its future maintenance costs. Because software development has strong influence on software maintenance, we discuss what may be done during development to reduce eventual maintenance. The actions during development may be taken with either in-house or contractor-developed software.

While this checklist is only an interim document, we feel that it will be useful to persons involved with computer software maintenance. The levels of effort and emphasis devoted to specific items mentioned will vary with the type and size of specific applications.

RECORDING THE MAINTENANCE WORKLOAD

Recording the maintenance workload at the installation identifies the types of maintenance being done such as where the maintenance dollars are being spent. For recording purposes, software maintenance should be subdivided into categories which are coded for later data reduction and analysis. Persons doing software maintenance should be required to report their work according to standard codes for software maintenance categories.

When NBS issues a standard definition of software maintenance and its component categories, that standard should be used. Of course, some parts of such a general Government-wide definition would not apply at individual installations.

As an interim set of categories for recording maintenance costs, pending publication of a detailed standard definition by NBS, we suggest the following six categories as a minimum level of detail for recording.

(1) Modify or enhance software to make it do new things for the end user that were not requested in the original system design.

(2) Modify or enhance software to make it do things for the end users that were called for in the original design but which were not present in the first production version of the software.

(3) Remove defects in which the software does something other than what the user wanted ("does the wrong things").

(4) Remove defects in which the software is programmed incorrectly ("does the desired calculation, but gives an incorrect answer").

(5) Optimize the software to reduce the machine costs of running it, leaving its user results unchanged.

(6) Make miscellaneous modifications, such as those needed to interface with new release of operating systems.

Staff time and computer costs for each of the six categories should be logged for each application program on which maintenance is done. A simple recording scheme suitable for later automated data reduction should be used. One method to record costs could be as follows.

(1) Adopt the standard definition of software maintenance recommended above.

(2) Using this definition, classify and code all activities and subactivities involved in the maintenance process being performed by data processing personnel.

(3) Assign codes to

- each user application,
- each employee, and
- each type of maintenance action.

(4) Where possible, use existing timekeeping and machine accounting procedures to report time spent on all software maintenance actions. The above codes should be used to identify the nature of each maintenance action. Where no time reporting system exists, a simple data gathering mechanism should be implemented.

(5) Establish cost centers to accumulate the costs for all significant software maintenance activities and subactivities. Within each cost center, costs may be aggregated by area of management responsibility and work function. Accumulated costs may be assigned to the benefiting applications. Indirect and overhead costs should be distributed to each work function. (See GAO guidelines for accounting for automatic data processing costs, app. III, ref. 4.)

Such data can be summarized to allow installation management to identify which categories of maintenance cost the most at their particular installation. Such identification of high-cost categories will enable managers to focus their efforts on areas which are likely to afford significant reduction.

Data on the types of maintenance now being done are needed to trace maintenance costs to their causes, which in turn may allow those causes to be reduced or eliminated. Data are needed as a basis for either type of management action--efficiency (do the maintenance cheaper) or prevention (avoid doing the maintenance at all).

ANALYZING THE MAINTENANCE WORKLOAD

Analyzing an installation's maintenance workload has three main purposes--(1) determining which types of maintenance cost the most at that particular installation, (2) linking each maintenance category to causes or reasons, some of which may be preventable, and (3) providing a basis on which to select actions to reduce software maintenance costs.

To make such an analysis possible, maintenance actions must be coded and identified in sufficient detail to allow maintenance actions to be connected to their probable causes. For example, a maintenance action reported simply as "Defect Removal" would give very little indication of what caused the need for the action. If carried to the next level of detail, however, with the defect identified as, say, being due to faulty program logic, a basis for determining the cause has been established. Management may determine the level of detail for grouping maintenance actions which it feels is necessary to allow a meaningful analysis, identify causes, and select actions.

Management actions at a particular installation will depend upon the results of analysis and fall into two basic categories--(1) efficiency/effectiveness, which means actions that cause us to do the software maintenance we are now doing in a cheaper or better manner, or (2) prevention, which means actions that will eventually reduce or eliminate the need for so much software maintenance. Of the two, increased efficiency provides the greater opportunity for reducing resource consumption in the short run. First, some efficiency measures may be more procedure oriented than policy oriented, and thus can be implemented at lower levels of management without lengthy policy review. Second, some efficiency measures are not as dependent on changes in related processes, such as system development, as is prevention.

To assist data processing managers in tracking causes of maintenance and selecting actions to take, we have provided two lists. One list concerns existing software, the other concerns future software and shows maintenance work types, causes, and possible preventive measures.

SUGGESTED STEPS TO INCREASE EFFICIENCY IN THE SOFTWARE MAINTENANCE OPERATION

The objective here is to increase the efficiency of maintaining existing software. Lessons learned during the maintenance of existing software can also provide feedback to development of new software so that excessive future maintenance costs can be avoided. Listed below are suggested steps that should be considered by data processing managers for their potential to increase efficiency in the software maintenance operation. These efficiency measures cover (1) general management procedures, (2) the maintenance operation, and (3) the development process.

I. General Management Procedures

--Establish criteria to measure efficiency in the maintenance standards and goals to manage by. After carefully analyzing the current maintenance workload, set standards which reflect the percentage of resources that management considers reasonable to maintain the current inventory of software. Levels of resources above these standards would be subject to management attention and subsequent action. Maintenance goals should reflect a lower level of resources expected to be attained by increased efficiency and by the use of techniques to reduce the necessity for future maintenance.

II. The Maintenance Operation

--Establish review procedures to properly evaluate the impact of maintenance actions on other parts of the software application prior to implementation. Improperly designed maintenance actions may adversely affect other processes and ultimately cause more defects which require additional maintenance to correct.

--Monitor individual programmer performance and, if deficiencies are found, provide training in the following areas:

(1) Time spent to perform maintenance tasks.

(2) Frequency of errors in program code causing additional maintenance.

(3) Ability to write clear, maintainable, well documented code easily understood by other programmers.

--Avoid random excessive use of "quick fixes" and patches to production systems. Efficiency may be improved by grouping well planned, carefully developed, fully tested, well documented changes to software applications and implementing them at scheduled intervals.

--Where possible, apply the principles of structured programming in maintaining existing programs. We are aware that many programs now running in production--and being maintained--were written before the advent of structured programming and that time does not permit a complete redesign to improve structure. However, structured programming can be followed locally within individual paragraphs or sections (COBOL) or subroutines (COBOL or FORTRAN) that are added during maintenance even though the programs they are added to are not structured throughout.

--Use structured programming in both development and maintenance work for more error-free programs, easier (less costly) maintenance, more efficient debugging, and easier-to-use documentation.

--Use labor saving aids whenever feasible including

(1) interactive terminals,

(2) on-line text editors,

(3) word processing or text processing systems to produce and update documentation, and

(4) software tools including code auditors, flow chart packages, test data generators, and program analysis tools.

--Ensure that all documentation is updated when maintenance is performed on a software application, so that future maintenance programmers will have a complete set of code, JCL, and test data to work with.

--Attempt to provide a work environment wherein program maintenance is not viewed by programmers as a dead end job with a stigma attached.

III. The Development Process

Ease of maintenance should be a prime consideration when computer programs are originally designed and developed. Programs can be written so that identification and correction of errors and expansion of the program to meet new requirements are easier, thereby reducing the time and effort required for all future maintenance. Some steps which management should require for more efficiency in later maintenance efforts are as follows.

--Select a programming language that is:

(1) Suitable for and widely used in the general area of the application. (For example, COBOL should not be used for

statistical applications because (a) it is not often used for that purpose, meaning that previous experience and published code are seldom available, and (b) it was not designed for such applications.)

(2) Widely available, widely known, widely taught, and likely to remain so for several years. The language should be available on different brands of computers. It should be widely known and widely taught so that future maintenance programmers who already know the language can be hired easily.

(3) Available from several vendors in well-tested production compilers. If there is a Federal standard for the language, the compiler used to develop the software should be one that has been validated by the GSA compiler test center.

--Use modular design and structured coding for the actual writing of the software. This must be expressed in a formal coding practices standards document and should be accompanied by inspection for compliance. The labor of such inspection can be greatly reduced by the use of automated aids which are widely available--some built into commercially available compilers, others embodied in separate software tools. If such inspection will be done, development personnel must be put on notice at the outset of the project that it will be done. Also, such inspection should be done during the development as well as at the end. Require documentation to be embedded in the source code, for example, code structure, meaningful data names, meaningful and truthful comments, and indentation conventions. Establish a group of minimum requirements for embedded documentation (for example, a standard preamble of comments) and require that this minimum be in the programs before they are submitted for their first compilation. Such requirements are greatly helped by productivity aids such as interactive text editors. Also, a number of software tools are available which will automatically reformat the source code of programs thereby reducing the labor of standardizing formats.

--Require external documentation as appropriate (FIPS PUB 38 is a reference). 1/ External documentation should include file and record layouts and prose about trade-offs between alternatives, assumptions made, and any deviations from language standards. Charts showing flow of data and control are also useful; the use of structured programming and higher level languages has decreased the value of the traditional flowchart.

1/App. III, ref. 8.

--Avoid the use of vendor-unique, CPU-peculiar 1/ or device-peculiar properties in constructing the software.

--Follow programming language standards where appropriate. If there is a national standard for the programming language being used, the software should be inspected for compliance with that standard. We are aware that use of vendor-unique extensions is sometimes justified, for example, by significant reduction in machine costs to run the programs or by reduction in data storage space needed. However, such extensions should be used only where a clear justification exists and such use and justification should be well documented.

With the COBOL language, compilers sold to the Government have been required for several years to include a "FIPS flagger"--an option which issues messages when nonstandard language extensions are used. 2/ When COBOL is the language used, programs should be inspected with the flagger option and unjustified extensions should be replaced with standard language.

--Use other design features for ease of maintenance, including:

(1) Limited number of interfaces between modules. 3/

(2) Communication between modules limited to the defined interfaces.

(3) Well-documented, easy-to-understand design.

(4) Limited equipment interfaces.

(5) A controlled data base.

(6) Limited access to the data base by each module.

(7) Programming style with clarity of function (readability and ease of verification).

(8) Usually one function per module, which leads to small modules.

(9) Separate modules for input, output, and computation of functions.

1/CPU: central processing unit.

2/FGMSD-80-4, op. cit. (See p. 18.)

3/See app. III, ref. 26.

STEPS TO REDUCE FUTURE MAINTENANCE

The objective here is to take actions during the development of new software to reduce its future maintenance costs. These actions will, and we believe should, be guided to a great extent by what the installation learns through analyzing the maintenance of its current software. Since software development has strong influence on software maintenance, we discuss what may be done during development to reduce eventual maintenance. The actions during development may be taken with either in-house or contractor developments. We recognize that a certain amount of maintenance will always be necessary. For this reason, we have limited our "probable causes" section of the checklist to what we consider "preventable" causes.

Type of maintenance action

Removing defects due to faulty program logic.

Probable causes

--Overly complex programs.

--Inadequate testing.

--New defects introduced through improper maintenance actions.

--Lack of proper change control on large development projects.

Prevention techniques

--Use certain software tools and techniques such as:

(1) Test data generators. These analyze a program and produce files of data needed to test the logic of the program.

(2) Structured programming (also called structured coding). A technique of developing computer programs so that they will be more easily understood by others who must later maintain and modify them. Such easier understanding aids documentation, testing, and correction.

(3) Top-down program development. This is the approach of designing, coding, and testing systems by building program modules starting with those at the general level (the "top") and proceeding down to the most specialized, detailed level (the "bottom").

--Adequately test computer programs using the following procedures:

(1) Test individual computer programs with test data that exercise the great majority--preferably 100 percent--of the procedural code of the programs. Tools exist which aid in demonstrating that logic was executed. Some are built into compilers as compilation options; others are separate software tools. The test data sets (and their outputs) used should be kept for later retesting in the production phase after user-requested changes have been made to the software.

(2) The software must be tested in as realistic a user scenario as possible, with functional review by the user as well as technical review by developers or separate quality control. When the software consists of a group of programs and files which must operate together to serve the user, the group must be tested as a group--individual tests of the parts are not usually enough. Interactive or other time-dependent aspects should be included in the testing.

(3) Auxiliary equipment, such as interactive terminals that will work with the software, should be included in the user test scenarios.

--Test maintenance changes, especially new releases, with appropriate test data sets kept from the testing phase.

--Adhere to installation programming practices standards during maintenance as well as during development. The idea here is that we do not want the software to "deteriorate" as time goes on--for example, to become less well structured, or to deviate from its documentation.

--Use formal change control procedures during development of software. Make sure all changes have been made in each stage of design prior to moving into the next stage.

--Review each proposed maintenance action for possible adverse impact on other portions of the software applications.

Type of maintenance action

Removing defects where program does something other than what the user wanted.

Probable causes

--Little or no user involvement in the development process.

--Faulty design and/or functional specifications.

--Misinterpretation of specifications by the programmer.

Prevention techniques

--Require user participation in the software development process both for the initial identification of user requirements and for user reviews of system output during the development process.

--Require independent inspection of software by someone other than the developer. This improves software quality by imposing discipline on the developer. It is now feasible to require such inspection because current tools can automate much of the work involved. Inspections should trace design specifications back to functional requirements to ensure those requirements are satisfied.

--Inspect software for functional completeness by asking such questions as:

(1) Does it do the user tasks (for example, calculations) that it was originally intended to do?

(2) Does it do those tasks correctly?

(3) Are its user outputs (printouts, etc.) and user documentation understandable to user representatives who have had no previous exposure to the development?

--State specifications in formal or quantitative terms rather than narrative English, so they are less likely to be misinterpreted.

--Utilize "top-down" design. (State requirements first in terms of general functions and then reduce them to module and program levels.)

Type of maintenance action

Fine tuning the software (optimization).

Probable causes

--Lack of performance criteria in the design stage.

--Inefficient coding practices.

--Failure to adopt and enforce programming standards.

Prevention techniques

--Augment functional specifications and requirements by requiring software to have quantitative performance attributes. In other words, specify not only what the software should do, but how efficiently it should do it.

We believe that machine efficiency can and should be addressed during the development phase for programs and systems which are expected to run for a long time and process a great deal of data. Of course:

(1) Correctness is more important than cheap operation.

(2) Many programs, such as data reduction programs, are not run enough times to justify exhaustive efficiency work. However, for programs expected to be expensive and longlived, efficiency should be addressed during construction because that is the best time to do it.

--During program development, use such software tools as program analyzers. This tool monitors the operation of an applications program and provides information used to make the program cost less to run on the computer.

--As a management tool, use preprocessors known as code auditors to determine whether programmers are complying with standards.

--Inspect software for compliance with programming practices standards, if the organization has such standards. Unjustified deviations should be corrected.

--Orient programmers toward efficient programming practices so they will write programs that are reasonably efficient and do not need later tuning. We believe that structured programming and machine efficiency do not conflict and indeed can complement one another. Orientation can be pursued through in-house seminars and pocket guides.

Type of maintenance action

Modifying software to make it do more end-user functions.

Probable causes

--Inadequate user requirement specifications in the design stage.

--Lack of user participation in the development process, both in the initial definition of requirements and in requirements design review.

--Inadequate change control during the development stage.

--Failure to test code for conformation to specifications.

--Inadequate management review procedures to determine the feasibility and necessity for user-requested changes.

--No fiscal liability to users for requested software changes.

Prevention techniques

--Require user participation in the software development process both for the initial identification of user requirements and for user reviews of system output during the development process.

--Make independent inspection of software mandatory by someone other than the developer. This improves software quality by imposing discipline on the developer. It is now feasible to require such inspection because current tools can automate much of the work involved. Inspections should trace design specifications back to functional requirements to ensure that those requirements are satisfied.

--Inspect software for functional completeness by asking such questions as:

- Does it do the user tasks (calculations, etc.) that it was originally intended to do?
- Does it do those tasks correctly?
- Are its user outputs (printouts, etc.) and user documentation understandable to user representatives who have had no previous exposure to the development?

--State specifications in formal or quantitative terms rather than narrative English, so they are less likely to be misinterpreted.

--Use "top-down" design. State requirements first in terms of general functions and then reduce them to module and program levels.

--Consider recovery of costs for modification efforts from the using organization to create more cost awareness among users. This would tend to eliminate some requests for "nice to have" enhancements.

SUMMARY RESULTS FROM OUR QUESTIONNAIRE

1. Does your installation have responsibility for any applications software maintenance as defined above? (If no, please explain below and skip to question 6.)

		Response count	Percent
(1)	Yes	357	87.3
(2)	No	51	12.5
(3)	No answer	1	.2
	Total	409	100.0

2. Which of the following best describes the extent of your installation's software maintenance responsibilities? (Please check only one.)

		Response count	Percent
(1)	Limited - Consists solely of identifying defects, new user requirements, troubleshooting, and installing changes for applications developed by a central agency function outside this installation.	24	5.9
(2)	Limited maintenance on centrally developed applications, plus maintenance on some locally developed systems.	137	33.5
(3)	Full maintenance responsibility for all applications run at this installation.	143	35.0
(4)	Other	54	13.2
(5)	No answer	a/ 51	12.5
	Total	409	b/ 100.1

a/Questions 2 thru 5 should have 51 "no answers"--the respondents who said no in question 1.

b/Should not add up to 100.

3. Is the applications software maintenance at your installation performed by installation employees, by contractor employees, or by a mixture of both? (Please check only one.)

		Response count	Percent
(1)	Installation employees do all applications software maintenance	252	61.6
(2)	Contractor employees do all the applications software maintenance	4	1.0
(3)	Applications software is maintained by a mixture of installation employees and contractor employees	86	21.0
(4)	Other (Please describe.)	14	3.4
(5)	No answer	53	13.0
	Total	409	100.0

4. Please show the percentage of total software maintenance (as measured by staff-hours) performed at your installation that falls in each of the following categories. Show a percent for each category (even if it is zero percent). Percents shown should add to 100 percent.

Any actions taken after implementation of the software to:

		Response count	Average percent reported
(1)	Remove defects in the software, including:		
	a. Defects in which the software was programmed to do something other than what the user wanted.	355	7.8
	b. Defects in which the program logic was faulty with the result that the program did something other than what the programmer intended.	355	11.1
(2)	Tune the software to make it more efficient (less machine time and/or less core).	355	11.1

	Response count	Percent
(3) Modify or enhance the software to make it perform more end-user functions, including:		
a. Functions originally called for in the system design, but not implemented.	355	9.2
b. New functions requested by the user not called for in the original system design.	355	41.5
(4) Make other modifications resulting from miscellaneous causes such as the need to interface with other systems, system software changes, etc.	355	19.2
Total		a/ 99.9

a/Not exactly 100 due to rounding.

5. Please estimate the percentages of the following resources' times that are devoted to the software maintenance functions listed in question 4. Please show a percentage for each item even if it is zero.

	Response count	Average percent reported (note a)
(1) Personnel		
Programmer/Analyst	353	52.9
Operations personnel	352	8.9
Administrative personnel	352	5.9
Management personnel	352	10.0
(2) Hardware		
CPU time	348	13.6

a/Should not add up to 100.

Part II--Installation Programming Information

6. Are the applications programs in use at your installation primarily business applications, primarily scientific applications, or a mixture of both business and scientific applications? (Please check only one.)

		Response count	Percent
(1)	Primarily business applications	219	53.5
(2)	Primarily scientific applications	65	15.9
(3)	A mixture of business and scientific applications	102	24.9
(4)	Other (Please describe.)	22	5.4
(5)	No answer	1	.2
	Total	409	a/ 99.9

a/Should not add up to 100.

7. Do most of the application programs in use at your installation run in production for a year or more before being discarded or replaced; do most run for less than a year; is the number running for a year or more about equal to the number that run for less than a year? (Please check only one.)

		Response count	Percent
(1)	Most run for a year or more	344	84.1
(2)	Most run for less than a year	23	5.6
(3)	About as many last a year or more as last less than a year	35	8.6
(4)	No answer	7	1.7
	Total	409	100.0

8. Next we are interested in the programming languages used in your installation. Please state for each of the languages listed below the number of application programs in that language currently in use in your installation and the average production life in years at your installation of the application programs in that language.

Language	Installations with programs in the language	Average no. of programs whose source code is in the language	Installations with over 100 programs in language	Average reported average length of production life of programs in the language in years
COBOL	263	746	170	5.4
FORTRAN	212	260	83	4.8
PL/I	42	128	7	4.0
BASIC	82	71	13	3.2
ALGOL	4	177	2	3.5
ASSEMBLY (note a)	197	215	71	5.8
CMS-1/CMS-2	5	81	1	2.2
PASCAL	6	20	-	1.5
RPG	41	83	12	3.3
LISP	-	(b)	(b)	(b)
SIMSCRIPT	10	9	-	3.1
GPSS	11	17	-	3.0
DYNAMO	2	8	-	3.0
SNOBOL	5	7	-	2.8
SCORE	6	45	1	2.3
EASY TRIEVE	23	84	3	2.4
DYL-260	7	110	2	1.7
DATA BASE LANGUAGES	84	67	16	6.0
OTHER (Specify)	64	173	16	3.4

a/Assembly languages include BAL, EASYCODER, AUTOCODER, GMAP, COMPASS, etc.

b/Not applicable.

9. Please write in the age (i.e., how long it has been in production) of the oldest application program in use at your installation. Write in your estimate.

Average (mean)	9.4 years
Respondents answering question	399

10. In what language is the oldest application program in use at your installation written? (Please check only one.)

(1) COBOL (2) Assembly language (Assembly languages include BAL, EASYCODER, AUTOCODER, GMAP, COMPASS, etc.)

(3) FORTRAN (4) Other (Please specify.)

	Response count	Percent
COBOL	167	40.8
Assembly language	85	20.8
FORTRAN	102	24.9
Other	45	11.0
No answer	10	2.4
Total	409	a/ 99.9

a/Not exactly 100 due to rounding.

11. Which, if any, of the following tools and techniques are in use at your installation? (Please check all that apply.)

	Tool	Responses having		Technique	Responses having
(1)	Automated documentation	105	(1)	Code arrangement	112
(2)	Source text manipulation	168	(2)	Descriptive documentation	219
(3)	Program optimization	131	(3)	Performance documentation	96
(4)	Aids built into compilers	199	(4)	Embedded documentation	205
(5)	Special programming languages compilers	89	(5)	Programming practices standards	200
(6)	Preprocessors	80	(6)	Reuse of already written code	262
(7)	Program performance evaluation	88	(7)	Quality assurance organization/ management	78
(8)	Design language	24	(8)	Design	149
			(9)	Programming organization/ management	134

Respondents answering question: 409

12. Does your installation have an ongoing (regular basis) effort to do optimization on application programs to reduce the machine costs of running them? (Please check only one.)

		Response count	Percent
(1)	Yes	137	33.5
(2)	No	252	61.6
(3)	Not sure	15	3.7
(4)	No answer	5	1.2
	Total	409	100.0

13. Are cost accounting procedures in effect at your installation to capture personnel, hardware, and overhead cost associated with application software maintenance as defined in this questionnaire? (Please check only one.)

		Response count	Percent
(1)	Yes	125	30.6
(2)	No	260	63.6
(3)	Not sure	18	4.4
(4)	No answer	6	1.5
	Total	409	a/ 100.1

a/Not exactly 100 due to rounding.

14. If yes, are reports showing these costs regularly produced? (Please check only one.)

		Response count	Percent
(1)	Yes	104	25.4
(2)	No	52	12.7
(3)	Not applicable-- no procedure	141	34.5
(4)	No answer	112	27.4
	Total	409	100.0

Part III--Opinions and Views

15. Based upon your experience, do you believe that application software developed by contractors requires more or less maintenance than application software developed in-house? (Please check only one.)

	Response count	Percent
(1) Contractor-developed software requires more maintenance	159	38.9
(2) Contractor-developed software requires about the same amount of maintenance	89	21.8
(3) Contractor-developed software requires less maintenance	22	5.4
(4) No opinion	136	33.3
(5) No answer	3	.7
Total	409	a/ 100.1

a/Not exactly 100 due to rounding.

16. In your opinion, which, if any, of the following actions would result in the greatest reduction in the size of the Government's applications software maintenance effort? (Please check only one.)

	Response count	Percent
(1) Better definition of user requirements in the system development stage	171	41.8
(2) Better definition of user requirements for modifications to existing software	13	3.2
(3) Use of software tools and techniques in system development (structured design, structured coding, etc.)	22	5.4
(4) Providing better tools and techniques for maintenance programmers (such as interactive terminals, text editors, and program analysis tools, etc.)	46	11.2

		Response count	Percent
(5)	More thorough testing of applications programs before the system is released to production	43	10.5
(6)	Eliminating unnecessary changes requested	12	2.9
(7)	Nothing--such a reduction is not possible by users	8	2.0
(8)	Other (Please specify.)	51	12.5
(9)	No answer	43	10.5
	Total	409	100.0

LIST OF SOFTWARE MAINTENANCE RELATED PUBLICATIONS

GENERAL ACCOUNTING OFFICE

1. "Wider Use of Better Computer Software Technology Can Improve Management Control and Reduce Costs," FGMSD-80-38, Apr. 29, 1980.

2. "Contracting For Computer Software Development--Serious Problems Require Management Attention to Avoid Wasting Additional Millions," FGMSD-80-4, Nov. 9, 1979.

3. "The Federal Information Processing Standards Program: Many Potential Benefits, Little Progress, and Many Problems," FGMSD-78-23, Apr. 19, 1978.

4. "Guidelines For Accounting For Automatic Data Processing Costs," Federal Government Accounting Pamphlet Number 4, 1978.

5. "Millions in Savings Possible in Converting Programs from One Computer to Another," FGMSD-77-34, Sept. 15, 1977.

6. "Improved Planning and Management of Information Systems Development Needed," LCD-74-118, Aug. 18, 1975.

NATIONAL BUREAU OF STANDARDS

7. "Guidelines for Documentation of Computer Programs and Automated Data Systems for the Initiation Phase," FIPS PUB 64, Aug. 1, 1979.

8. "Guidelines for Documentation of Computer Programs and Automated Data Systems," FIPS PUB 38, Feb. 15, 1976.

9. "Appraisal of Federal Government COBOL Standards and Software Management: Survey Results," by Donald R. Deutsch, NBSIR 76-1100, Aug. 1976.

10. "Computer Software Management: A Primer for Project Management and Quality Control," by Dennis W. Fife, NBS Special Publication 500-11, July 1977.

GENERAL SERVICES ADMINISTRATION

11. "Management Guidance for Developing and Installing an ADP Performance Management Program," Nov. 1978.

OTHER SOURCES

12. "A Procedure to Review and Improve the Operational Efficiency of Production Systems," by Robert Grossman, CDP, Proceedings, Joint ACM/NBS Symposium, June 1980.

13. "Certification Testing: A Procedure to Improve the Quality of Software Testing," by Alfred R. Sorkowitz, Computer, Vol. 12, No. 8, Aug. 1979, pp. 20-24.

14. "Programmer's Quick References for COBOL Structured Programming," HUD-544-2-ADP, Nov. 1979.

15. "A Review of Software Maintenance Technology," RADC-TR-80-13, by John D. Donahoo and Dorothy Swearinger, Rome Air Development Center, Feb. 1980.

16. "The New Software Economics," by Werner L. Frank, Prentice-Hall, 1979.

17. "The World of Software Maintenance," by Girish Parikh, Computer World, 1979.

18. "Summary of Findings: A Critical Assessment of EDP Objectives," McCaffery, Seligman, and Von Simson, Inc.; Sept. 1978.

19. "Standard Definitions: A Missing and Needed Software Tool," by Steven Merritt, Proceedings, 17th annual ACM/NBS Symposium, Gaithersburg, Md., June 15, 1978.

20. "Programmer's Guide: 5.0 Programming Standards," U.S. Customs Service, AMPS Program Division, June 1, 1978.

21. "The Software Life Cycle - A Management and Technological Challenge in the Department of Defense," by Barry C. De Roze and Thomas H. Nyman, IEEE Transactions on Software Engineering, Vol. SE-4, No. 4, July 1978, pp. 309-318.

22. "Corporate-Level Software Management," by John D. Cooper, IEEE Transactions on Software Engineering, Vol. SE-4, July 1978, pp. 319-326.

23. "Controlling the Software Life Cycle--The Project Management Task," by William C. Cave and Alan B. Salisbury, IEEE Transactions on Software Engineering, Vol. SE-4, No. 4, July 1978 pp. 326-334.

24. "Managing the Maintenance Programming Function," (14-05-01) Computer Programming Management, Averbach, 1978.

25. "Characteristics of Application Software Maintenance," by B.P. Lientz, E.B. Swanson, and G.E. Tompkins; UCLA, 1977.

26. "Software Acquisition Management Guidebook: Software Maintenance," by J.R. Stanfield and A.M. Skrukrud, ESD Hanscom AFB, Mass., Oct. 1977.

27. "Software Acquisition Management Guidebook: Software Development and Maintenance Facilities," MITRE, ESD Hanscom AFB, Mass., Apr. 1977.

28. "A Look at Software Maintenance," by Chester C. Liu, Datamation, Nov. 1976, pp. 51-55.

29. "The Mythical Man-Month," by F.P. Brooks, Jr., Addison-Wesley Publishing Company, 1975.

30. "Application Program Change Control," Regulation No. 18-21-1, Department of the Army, July 1, 1975.

31. "Improved Application Development Pays Off at Marathon Oil," Infosystems, Apr. 1975, p. 10.

32. "An Overview of Programming Practices," by J.M. Yohe, University of Wisconsin--Madison Mathematics Research Center, May 1974.

33. "The Effect of Software Structure on Software Reliability, Modifiability, and Reusability: A Case Study and Analysis," by John B. Goodenough, et al.; Frankford Arsenal, July 1974.

34. "Improved Programming Technologies: Management Overview," IBM, 1973.

35. "Software Life Cycle Cost Considerations," by Barry, Nichols, and Schiff; IBM/FSD, Gaithersburg, Md., 1973.

36. "That Maintenance Iceberg," EDP Analyzer, Oct. 1972, Vol. 10, No. 10.

37. The Art of Software Testing, by Glenford J. Myers, (N.Y.: Wiley, 1979).

38. Implications of Using Modular Programming, Hoskyns (London: H. M. Stationery Office, 1973).

39. "Auditing Computers With A Test Deck," General Accounting Office, 1975.

40. "Software Testing in Computer-Driven Systems," by J. Gary Nelson, Software Quality Management, ed. J.D. Cooper (N.Y.: Petrocelli, 1979), ch. 16.

41. "Real Time: The Lost World of Software Debugging and Testing," by Robert L. Glass, Comm. ACM, May 1980, Vol. 23, No. 5.

42. "Quality Assurance Tools," by Robert W. Shirey, Computer-world, May 19, 1980.

43. "The Structure of Modular Programs," by Joshua Turner, Comm. ACM, May 1980, Vol. 23, No. 5.

44. "EDP Performance Management Handbook, Vol. II Tools and Techniques," Applied Computer Research.

45. "Optimizing Program Quality and Programmer Productivity," by Capers Jones, IBM, SHARE 50, Mar. 7, 1978.

46. "Tackle Software With Modular Programming," by John Rhodes, Computer Decisions, Oct. 1973.

47. "The Search for Software Reliability," EDP Analyzer, May 1974, Vol. 12, No. 5.

48. "Embedded Computers: Software Cost Considerations," by John H. Manley, AFIPS, NCC 1974.

49. "Software Life Cycle Management," by John H. Manley, Gaithersburg, Md., Aug. 1978.

SITES AT WHICH WE ANALYZED SOFTWARE MAINTENANCE IN DETAIL

1. U.S. Railroad Retirement Board, Chicago, Illinois
2. Argonne National Laboratory, 9700 South Cass Ave., Argonne, Illinois
3. Veterans Administration Data Processing Center, Austin, Texas
4. Air Force Manpower and Personnel Center, Randolph AFB, Texas
5. Bureau of Reclamation, Water and Power Resources Service, Denver, Colorado
6. Veterans Administration Data Processing Center, Hines, Illinois
7. San Antonio Air Logistics Center, Kelly AFB, San Antonio, Texas
8. Air Force Accounting and Finance Center, Directorate of Data Automation, Denver, Colorado
9. U.S. Army Troop Support and Aviation, Material Readiness Command, 4300 Goodfellow, St. Louis, Missouri
10. Oklahoma City Air Logistics Center, Tinker AFB, Oklahoma City, Oklahoma
11. Bureau of the Mint, Mint Data Center, San Francisco, California
12. 552d Airborne Warning and Control Wing, Tinker AFB, Oklahoma City, Oklahoma
13. National Aeronautics and Space Administration, AMES Research Center, Moffett Field, California
14. United States Postal Service, Postal Data Center, St. Louis, Missouri
15. United States Postal Service, Postal Data Center, San Bruno, California

APPENDIX V

General
Services
Administration Washington, DC 20405

Honorable Elmer B. Staats
Comptroller General of the United States
U.S. General Accounting Office
Washington, DC 20548

Dear Mr. Staats:

We have reviewed the General Accounting Office draft report entitled "Software Maintenance: Expensive and Undermanaged," dated October 21, 1980, and we agree that software maintenance is a high-cost area which to date has not received adequate management attention.

The rising cost of all software activities requires more effective and efficient management. We believe that software maintenance should be considered as a unique element in the context of the overall software management problem. Poor and costly software maintenance is a direct result of poor software management and requires no new, unique solution, but rather the application of good software management techniques.

The brief discussion in the draft report concerning the definition of "software maintenance" illustrates this point. We believe that the data processing community understands the term as you have defined it. The fact that the term "tuning the software to make it more efficient and economical to operate" includes two elements very much akin to software development and conversion illustrates that software activities must be managed as an entity.

It is in response to the previous lack of attention to the software area that General Services Administration (GSA) recently established the Office of Software Development to provide a specialized center of software expertise for agency assistance. The office's planned program activities are designed to assist agencies in reducing software costs and improving software productivity. Software maintenance is one important element which these activities will address.

Although the report's recommendations are directed to the National Bureau of Standards (NBS), GSA plans to assist NBS in any way possible to provide guidance to Federal agencies concerning software maintenance.

APPENDIX V

We suggest that the following points be emphasized in the final report:

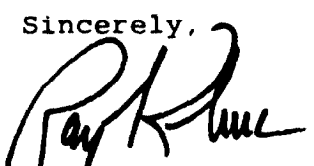
1. The need for software product modifications can be minimized by proper management of the software development process through (a) the informed participation of management in the software development and software maintenance decision making process with regard to the definition of software requirements to meet agency needs, (b) ensuring that user needs are met by the basic system through close coordination with the user and participation of the user in the development process, and (c) ensuring that the user understands the need for a cutoff of changes in requirements during the development process, in order to provide a workable product.

2. Software maintenance can be facilitated by the use of (a) good practice in design and programming of the software products, currently known as "structured analysis, design, and programming," and (b) insistence on the production of detailed and accurate documentation as an integral part of the software development. We recommend that NBS explore standardization in these areas.

3. When software development is contracted for, appropriate contracting techniques, such as fixed-price contracting and payment based upon acceptance of deliverable products, can minimize software maintenance efforts. GSA will soon publish an FPR/FPMR bulletin entitled "Software Development Contracting Guidelines," responsive to GAO Report FGMSD-80-4, dated November 9, 1979. The intent of the guidelines is to assist agencies in contracting for software products that can be maintained more efficiently and economically.

The opportunity to comment on this draft report is appreciated.

Sincerely,



K. C.
Admin.

APPENDIX V

THE POSTMASTER GENERAL
Washington, DC 20260

November 14, 1980

Dear Mr. Anderson:

This refers to your proposed report entitled "Software Maintenance: Expensive and Undermanaged."

The Postal Service agrees with the report's overall recommendations and already has measures underway in keeping with your recommendations to the heads of agencies:

1. Manage software maintenance as a discrete function.

   We assign software maintenance for specific ADP systems to specific Data Processing Centers and have established centralized control at USPS Headquarters over maintenance requests for all national systems. We are also installing detailed maintenance management procedures that prescribe a life-cycle methodology for maintenance projects of all sizes.

2. Identify the resources spent on software maintenance.

   This will be addressed as an element of our new maintenance management procedures.

3. Develop maintenance standards and goals.

   We are now measuring and analyzing our maintenance workloads and by early 1981 we will have developed standards and goals.

APPENDIX V

4. Increase the efficiency of the maintenance operation and reduce future maintenance.

   Our new maintenance management system will accomplish these goals.

Thank you for the opportunity to comment on your fine report.

Sincerely,

William F. Bolger

Mr. William J. Anderson
Director, General
Government Division
United States General
Accounting Office
Washington, D.C. 20548

APPENDIX V

National Aeronautics and
Space Administration

Washington, D.C.
20546

Reply to Attn of L

NOV 1. 1980

Mr. W. H. Sheley, Jr.
Acting Director
Procurement and Systems
Acquisition Division
U.S. General Accounting Office
Washington, DC 20548

Dear Mr. Sheley:

Thank you for the opportunity to review GAO's draft report entitled, "Software Maintenance: Expensive and Undermanaged," (Code 913500), which was forwarded with your letter dated October 15, 1980.

The draft has been reviewed by NASA staff at Headquarters and at the NASA Centers involved in the assignment. The comments on the GAO recommendations are provided in the enclosure.

If we can be of further assistance, please let me Know.

Sincerely,

Gerald D. Griffin
Associate Administrator
for External Relations

Enclosure

APPENDIX V

NOV 14 1980

Comments on GAO Proposed Draft Report Entitled, "Software Maintenance: Expensive and Undermanaged," Code 913600

NASA is in favor of voluntary guidelines for the management of software when the overall life cycle perspective is taken. We would encourage the development of such voluntary guidelines. But we have reservations with the approach taken by this draft GAO report.

Three principal concepts within the report concern us: 1) the definition of software maintenance is too broad and will lead to confusion of management responsibilities and costs; 2) research and development software is managed differently from production oriented administrative software and the report doesn't recognize that difference; 3) the advantage to isolating the software maintenance function from the overall life cycle management function is not apparent.

NASA believes the GAO definition of software maintenance is too broad and is apparently not alone in this belief. By including the phrase "doing more and different tasks" within the definition of software maintenance has created a potential problem. The definition causes an artificial shifting of management responsibilities, prerogatives and associated costs. In a Research and Development (R&D) environment modifying the software and making it do more and different things is the essence of the development cycle. As such the responsibility and the decision to modify the software should be the experimenter's decision not someone charged with maintaining production software. Secondly, these changes should appropriately be charged as development costs, not as maintenance. The proposed GAO definition confuses these issues.

There is a real difference in managing scientific software as compared to production oriented nondynamic code. Our concern is that GAO will develop generalized guidelines based on its sample of business oriented sites and this will create general purpose regulations. Regulations that by their very nature can not take into account the requirements of scientific software management. Research and Development agencies like NASA do not fit the mold generated by the GAO sample. The indiscriminate application of broad guidelines would prove to be counter productive and would impose unnecessary overhead and impact effective management.

The report assumes more management is needed based on the fact software costs are high. By itself this is not adequate justification. The government is not the exception, it is a high cost item within industry also.

One reason the cost is reported as so high in the GAO report is directly attributable to the broad definition of software maintenance. It includes items normally considered software development costs. The GAO report indicates 60 percent of the maintenance activity is software modification, that adds new and different tasks. This inflates the government maintenance cost. The effort to manage software maintenance discretely will also add to the maintenance overhead.

Discrete management forces the creation of artificial goals and limits to measure success. It also implies the creating of data gathering and processing systems to measure these artificial goals. This all contributes to more nonproductive overhead.

NASA uses the majority of its computers as research tools. Instead of setting artificial limitations as is suggested, management should judge its success by measuring user satisfaction and adhering to overall life cycle goals and objectives.

We feel that goals and objectives in life cycle management could be enhanced through voluntary guidelines. Therefore, we encourage their development within the framework we have discussed above.

E. C. Kilgore
Associate Administrator for
Management Operations

(913600)

AN EQUAL OPPORTUNITY EMPLOYER

UNITED STATES
GENERAL ACCOUNTING OFFICE
WASHINGTON, D.C. 20548

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE, $300

POSTAGE AND FEES PAID
U. S. GENERAL ACCOUNTING OFFICE

THIRD CLASS