

12554112172

BY THE COMPTROLLER GENERAL

# Report To The Congress

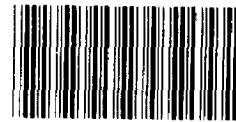
OF THE UNITED STATES

## Wider Use Of Better Computer Software Technology Can Improve Management Control And Reduce Costs

Because the Federal Government spends billions of dollars annually on computer programs, GAO made this review to determine if the latest tools and techniques were being used to develop and maintain those programs. Computer specialists at many agencies were unaware of the newer, better methods; others were reluctant to change to them. The Government can improve management control and save significant amounts through appropriate use of better software tools and techniques. GAO recommends various ways to promote such use.



010002



112172

FGMSD-80-38  
APRIL 29, 1980





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

B-115369

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 potential benefits that modern software tools and techniques offer for improving Federal agencies' management of software while reducing its costs.

We are sending copies of this report to the Director of the Office of Management and Budget, the Secretary of Commerce, and the Administrator of General Services.

*James B. Atch*  
Comptroller General  
of the United States



D I G E S T

Many Federal installations have not exploited the benefits of modern software tools and techniques as well as they could have. The only organizations which have consistently done so are those private firms which sell software.

A software tool is a computer program that aids the work of producing or maintaining other computer programs or their documentation. Software techniques include (1) technical methods or procedures which aid the production of, or improve the quality of, computer programs and their documentation and (2) management techniques which can improve the predictability and control of these activities.

During its work in automatic data processing, GAO found many areas where using better software tools and techniques might have saved considerable money and trouble. These include:

--Management control. (See p. 16.)

--User needs. (See p. 11.)

--Maintaining and modifying production computer programs. Programs cost as much or more to maintain and modify as they cost to develop, and software tools and techniques can reduce these costs. (See p. 21.)

--Software conversion. A recent GAO report stated that the annual cost of Federal conversions is estimated at over \$450 million. (See p. 11.)

One private firm estimated savings of \$1 million per year in the development and maintenance

FGMSD-80-38

of applications, the comprehensive use of better software tools and techniques including preprocessors and structured programming. At a military installation, officials estimated that over \$1 million in equivalent computer capacity was freed by software performance improvement--analyzing and modifying programs so that they will cost less to run on the computer while still giving the same answers. More can be saved by taking a software tool developed at one installation and using it at others.

### CONCLUSIONS

Federal use of software tools and techniques can be improved by better guidance to agencies, more emphasis on software by management, and effective Government-wide coordination and sharing of tools. Many opportunities exist for greater use of software tools and techniques in many Federal agencies. However, agencies' adoption of the newer technology should be based on a careful study of all costs and benefits. Unless Federal automatic data processing management makes more use of such technology, Federal computer software will continue to cost millions more than is necessary.

### RECOMMENDATIONS

The Director of the Office of Management and Budget should:

- Require heads of Federal agencies to establish software quality assurance functions in their agencies. In establishing such functions, heads of Federal agencies should promote modern practices for software that is developed for their agencies--both in-house and by contractors.
- More clearly define the responsibilities of agency heads and automatic data processing managers for the acquisition, management, and use of software tools and techniques.
- Direct the establishment of coordinated Government-wide research and development

for software tools and techniques which will include provision for disseminating information to all potential Federal users.

The Administrator of General Services should:

- Modify Federal Procurement Management Regulation 101-35.206 to incorporate actions agencies should take to improve their applications software, as suggested in appendix I.
- Establish a set of standard tools for solving operational problems and promoting efficiency and economy.
- Require that certain standard inspections, using software tools, be done on contractor-developed software.
- Establish a software tools category in the Federal Software Exchange Center and provide technical aid for the sharing of tools.

The National Bureau of Standards should develop or adopt standards or guidelines for using software tools.

#### AGENCY COMMENTS

In written comments on the report, the General Services Administration agreed with GAO's conclusions and recommendations. The Department of Commerce concurred with the intent of the report but made some suggestions for rewording. In the light of their suggestions, we clarified certain details of our presentation. The changes made were not substantive. The Office of Management and Budget agreed with GAO's conclusion that agencies are not always using the latest software technology, and said that it would support agency budget requests for new technology where justified, and would consider modifying Circular A-71. (See app. III and p. 40.)

2000

2001

2002



# C o n t e n t s

		<u>Page</u>
DIGEST		i
CHAPTER		
1	INTRODUCTION	1
	Software has become the most	
	costly part of computer systems	1
	Types of computer software	2
	Production applications programs	2
	Systems programs	4
	Utility programs	4
	An illustration	4
	Software tools	5
	Software techniques	6
	Roles of various agencies	7
	Congressional and Presidential	
	concerns	7
2	OPPORTUNITIES TO IMPROVE	
	COMPUTER SOFTWARE	9
	Few organizations have achieved	
	what is possible with better	
	tools and techniques	9
	Software technology significantly	
	exploited only by firms whose	
	product is software	9
	Other private firms--and some	
	Government installations--lag	
	in use of software technology	10
	Poor software is very costly	11
	Serious impact on user tasks	11
	Redundant software efforts	11
	Excessive software conversion	
	costs	11
	Excessive costs and frustra-	
	tions from a poorly designed	
	supply system	12
	Capacity of new computers lost	
	due to inefficient software	12
	Ineffective communications sys-	
	tem did not meet all user needs	
	and incurred time delays and	
	cost overruns	13
	Better software tools and techniques	
	would have reduced or avoided those	
	problems	13
	Productivity in software development	
	has shown little improvement over	
	the years	14

**CHAPTER****Page**

3	BETTER SOFTWARE TOOLS AND TECHNIQUES NOW AVAILABLE HAVE IMPROVED MANAGEMENT CONTROL AND SAVED MONEY WHEN USED	16
	Management control can be improved	16
	Software quality can be improved	17
	Costs of developing production applications software can be reduced	18
	Equipment procurements can be deferred	20
	Cost of maintaining, modifying, converting, and operating pro- duction applications software can be reduced	21
	Productivity can be raised by avoiding redundant developments	23
	Our experiments verified the potential of reducing software costs	25
4	WHY MODERN SOFTWARE TOOLS AND TECH- NIQUES HAVE NOT BEEN MORE WIDELY USED IN THE GOVERNMENT	27
	Lack of Government-wide policy, guidance, and standards	27
	Agency managers often fail to require inspection of contractor- developed software	29
	Neither management nor independent review groups emphasize software improvement	30
	Other reasons some installations use new tools and techniques so little	31
	Poor sharing of tools within and between agencies	33
	Lack of coordinated research and development	34
5	CONCLUSIONS, RECOMMENDATIONS, AND AGENCY COMMENTS	36
	Conclusions	36
	Recommendations	37
	Agency comments	39
	General Services Administration	39
	Office of Management and Budget	40
	Department of Commerce	40
6	SCOPE OF REVIEW	42

APPENDIX		<u>Page</u>
I	Suggested augmentation of Federal Procurement Management Regulation 101-35.206	43
II	List of software-related Government publications	44
III	Agency comments	48
IV	Detailed response to Commerce comments	54

#### ABBREVIATIONS

ADP	automatic data processing
COBOL	common business-oriented language
DBMS	data base management system
DOD	Department of Defense
FORTRAN	formula translation
GAO	General Accounting Office
GSA	General Services Administration
NBS	National Bureau of Standards
OMB	Office of Management and Budget

10/10/10

10/10/10

10/10/10

## CHAPTER 1

### INTRODUCTION

Today, computers are very important in our lives. They have greatly increased our abilities with technological advancements which were unthinkable 10 to 20 years ago. Software--the programs that make a computer run--is the most expensive component of computer services and is also the component with the greatest potential adverse impact on the user tasks.

#### SOFTWARE HAS BECOME THE MOST COSTLY PART OF COMPUTER SYSTEMS

The complexities and problems of writing and maintaining computer programs have caused software costs to outstrip computer hardware costs. Recent studies predict, and a prior GAO report notes, that by 1985 over 90 percent of the cost of automatic data processing (ADP) will be attributable to software. (See app. II, no. 8.) In the early 1970s, the Federal Government was spending over \$2 billion a year on software, and experts believe that today such expenditures exceed \$6 billion yearly. In 1977, the Federal accumulated investment in current software was estimated at over \$25 billion.

During the mid-1960s and early 1970s, some organizations realized that they had serious management and technical problems with software. Some of the management problems included a lack of ability to control software development projects, and high and unpredictable costs for software development and maintenance. The technical problems included programmers' time wasted on clerical tasks, difficulty in making given software run on several different computers, and difficulty in modifying existing software for changing user requirements. The organizations with these problems began to develop new software tools and techniques to reduce their problems and costs.

In this report, we define a software tool as a computer program that aids in producing or maintaining other computer programs or their documentation. A software technique is a method or procedure which reduces the work of producing or maintaining computer programs or their documentation, or which aids in managing these activities.

Now many tools and techniques have been developed which offer significantly improved management control and reduced costs if properly applied. Other tools and techniques are emerging.

This report discusses the Federal Government's potential benefits from the proper application of software tools and techniques. The report also presents the status of Federal and private sector software technology, and examples of savings and other benefits that have been achieved in both sectors.

## TYPES OF COMPUTER SOFTWARE

Computer software has been defined as the programs which make the computer run, the data files which those programs process, and explanatory material--called documentation--which accompanies the programs and files. Often, however, the word "software" is used to refer only to the programs. Computer programs generally can be grouped into applications programs, systems programs, and utility programs.

### Production applications programs

These programs automate the tasks of end users. For example, the checkwriting program of a payroll system writes checks for employees. The tasks automated for end users are almost endless: payroll, billing, and inventory in the business sector; simulations and statistical processing in the scientific sector; and air traffic control and satellite tracking in the command, control, and communications sector.

Applications programs have life cycles which can be divided into a development phase and an operational or production phase.

The development phase consists of:

- Requirements definition. Define, quantify, and document the needs of the end users for whom the software is to be written; for example, the detailed specifications of what is needed to achieve the general objective, "process payroll for 20,000 employees every 2 weeks."
- System design. Select the combination of computer programs, procedures, computers, people, and documentation needed to build and operate the system.
- Program design. Plan the individual computer programs that will actually process the data in accordance with the system design.
- Programming. The writing of the computer programs (coding).

--Testing. First running the individual programs against test data to see if the programs work properly; then testing the programs and procedures in combination to see if they accomplish the user's task.

The objective of applications software development is to construct computer programs that will process the users' data correctly at as low a cost as feasible, and document those programs so that they can easily be modified later if necessary. Software development is a labor-intensive, error prone process. Errors can be made both in deciding what the programs should do, and in constructing them to do it. Experience has shown that the longer errors go undetected as development progresses, the more expensive they are to correct. Ambiguous requirements definitions, or errors in design or programming which are undetected until the later testing or operational phases, can cause extensive rework or even abandonment of large projects.

The operational phase begins when the applications program produces its first user output and includes:

--Maintenance. Work done on the programs and/or their documentation after they are in production to correct errors and omissions. Many people use the term "maintenance" to include the work which we call "modification" below.

--Modification. Work done to make an existing system accomplish additional user requirements above and beyond those originally intended.

--Performance improvement. Work done on operational programs to make them cost less to operate. It is also called optimization.

--Conversion. Work done to make programs run on a computer other than the one for which they were originally written. It may occur during the operational phase of the life of applications software.

The work done on programs and documentation during their operational phase can account for up to 70 percent of total life cycle costs and consume most of an organization's computer programming labor. Experience has shown that continual changes to computer programs have often been necessary to correct hidden errors, to improve efficiency, or to add new user functions because of legal, administrative, or technical changes. The objectives of software maintenance and modification are to fix errors as soon as possible and to install needed user changes correctly with as little effort as

feasible. Maintenance and modification are very expensive and very hard to control.

Applications programs and documentation may be developed, maintained, modified, improved, or converted by employees of the user's organization or these activities may be done by specialist firms. Also, programs may be bought readymade from firms selling software built to serve many users. In some cases, firms which sell computers also will sell applications software.

### Systems programs

Systems programs automate the control of operation of the computer and auxiliary equipment. They control the running of applications programs and utilities (see below), control the allocation of systems resources to the programs, and report on the resources used to run the programs. Systems programs are usually supplied by the computer vendor, but may be obtained from other suppliers.

### Utility programs

Utility programs aid the tasks of computer programmers and others who work with the computer. They include language translators <sup>1/</sup> and stored routines for very common tasks such as sorting data. Utility programs may be supplied by the hardware vendor or independent software firms, or written by the user's employees. Software tools are a particular class of utility programs which aid work done on other computer programs.

### An illustration

Definitions are not standard in the software field--especially as to where systems programs end and utility programs begin. However, an example will illustrate our use of the terms. The checkwriting program of a payroll system is a production applications program which prints checks for end users. While the checkwriting program is running, its operation and its access to computer resources such as memory, disk, and tape, are controlled by the supervisory control module of the operating system--a systems program. The checkwriting program would be written with the aid of utilities,

---

<sup>1/</sup>Language translators are compilers and interpreters which transform the statements of programming languages written by humans into internal machine codes which directly control computers.



including a language translator. Also, the timecards containing the data processed by the checkwriting program may have been presorted for processing by another utility program--the .sort/merge.

Our report concerns the use of modern software tools and techniques to alleviate the problems of the development, maintenance, modification, operation, and conversion of applications software.

## SOFTWARE TOOLS

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 variety of tools is available commercially and at Federal agencies. These tools range in size and complexity from simple aids for individual programmers to complex tools that can support many software projects at the same time. However, many of these tools are known only in the organizations which developed them. Instead of using a tool which already exists elsewhere, an organization may needlessly develop another. The following are some common tools.

- 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 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, key-punching, 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 applications 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

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. 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").
- 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. Concept of developing documentation concurrently with the development of a program 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 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.

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

--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.

The importance of better tools and techniques is shown by the research being conducted by organizations whose automated systems are vital. An example is the Department of Defense's (DOD's) effort to develop a completely new programming language--specifically for structured programming--for its tactical computers which are embedded in weapons systems. We are reviewing this DOD effort separately because of its size and cost.

#### ROLES OF VARIOUS AGENCIES

The basic law governing Federal ADP management is the Brooks Act, Public Law 89-306. Under this act, the General Services Administration (GSA) is responsible for procuring and maintaining Federal ADP resources. GSA receives technical advice from the Secretary of Commerce primarily through the National Bureau of Standards (NBS), and both Commerce and NBS receive fiscal and policy guidance from the Office of Management and Budget (OMB).

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

#### CONGRESSIONAL AND PRESIDENTIAL CONCERNS

On October 1, 1976, the House Committee on Government Operations issued a report on the administration of Public Law 89-306. This report stated that, through an agency's efficient use of ADP, noncompetitive interim upgrades, additions, and replacements can be minimized. OMB, GSA, and NBS were criticized for not providing overall leadership and guidance in this area. User agencies were criticized for not attempting optimum ADP use to achieve a high level of economy and efficiency in their programs.

In September 1977, the Chairman of the Research and Development Subcommittee of the House Committee on Armed

Services, in a letter to DOD's Director of Defense, Research and Engineering, stated:

"This Committee recognized and appreciates the benefits derived from digital processing. I similarly recognize that digital computation not only generates advantages such as flexibility, repeatability, etc., but disadvantages in terms of software development, software proliferation and software maintenance. Digital computers are being used in a growing number of applications throughout the Department of Defense; however, this Committee has observed a lack of a cohesive plan to address the Department of Defense software problem."

In July 1978, the Chairman of the House Committee on Government Operations made the following comments in a letter to the Comptroller General requesting a report on the use of software tools and techniques.

"\* \* \* the Committee has conducted numerous investigations \* \* \*. One of the most disturbing findings is that many Federal agencies are designing and developing software which is inefficient and does not fully take advantage of the many tools and techniques available to aid them in their efforts."

The President initiated the Federal Data Processing Reorganization Project study to improve the Government's use of information technology. This 1978 study points out some of the major deficiencies in applying software technology in the Federal Government, such as the need for (1) better training in the use of modern technology, (2) greater Federal research and development efforts to produce technology tailored to Government needs, (3) greater use of software tools and techniques, (4) better software standards programs, and (5) better Government-wide guidance.

## CHAPTER 2

### OPPORTUNITIES TO IMPROVE

#### FEDERAL COMPUTER SOFTWARE

Until recently, because software development was considered an art, it was allowed to grow under the control of ADP technicians rather than management. Significant problems resulted: development of software systems which cost too much to operate and maintain, cost overruns and delays in software development projects, redundant software developments, excessive software conversion costs, and delays when computer equipment was changed. We believe that better use of modern software tools and techniques would have reduced the costs and avoided some of the problems.

Software tools and techniques can be powerful aids in the design, development, test, and maintenance of computer software. We found several situations where the application of tools and techniques resulted in significant benefits including improved management control, equipment procurements which could be deferred, and reduced software costs.

Despite the potential for improvement in the software area, we found that many organizations--Federal and private--have not exploited what is available to them. We also found recurring software problems and costs.

#### FEW ORGANIZATIONS HAVE ACHIEVED WHAT IS POSSIBLE WITH BETTER TOOLS AND TECHNIQUES

We found that private firms which sell software or produce it on contract are the only organizations which consistently exploit the newer software technology. Most Government and private facilities have made only moderate use of better software tools and techniques.

#### Software technology significantly exploited only by firms whose product is software

We assessed current practices in using software tools and techniques through a combination of site visits, the results of our questionnaires, our interviews, and a literature search. We found it necessary to divide the private sector into firms whose product is software and other firms which use software, in-house developed or bought, but do not produce it for sale.

We found that most managers and staff members of software-producing firms were very interested in modern programming

practices and new technology. In fact, the software firms we visited used software tools and techniques extensively in day-to-day operations. Similarly, we found that private software firms have explored portability concepts in using tools for software development and maintenance on different brands of hardware. The private software firms also have extensively used quality assurance techniques and tools in auditing and inspecting software development, testing, and operation.

Other private firms--and some  
Government installations--lag  
in the use of software technology

Certain Government units and private firms other than software firms do use the newer, better software tools and techniques to achieve overall ADP efficiency and effectiveness. However, many such installations have achieved only limited to moderate progress in applying this technology--and some have not used it at all.

We found Government agencies to be somewhat aware of modern programming practices and interested in what these tools and techniques can accomplish in software development and maintenance. We also found that usage of modern programming practices at Government agencies varied widely, but the majority of those we contacted made limited use of these practices. Government facilities generally adopt modern software tools and techniques slowly, although we found some instances where good work was being done. The limited sharing of software tools and techniques between Government facilities has been done more informally than through the GSA Software Exchange Center. We found Federal installations' active participation and productive use of modern programming practices for quality assurance purposes to be more the exception than the rule.

In general, we found the private sector to be moderately aware of software tools and techniques, especially modern programming practices such as structured programming. However, private firms also displayed a wide range of usage and application of software tools and techniques. For the most part, the private firms, other than software producers, make limited to moderate use of software tools. However, we saw more interest in, and more organizational effort devoted to better use of, software techniques at the private sector facilities we visited. We saw several impressive uses of quality control methods to improve software development at private firms, but most of these were software producers. Concerning sharing of tools, the private sector has a profit motive and little incentive to share with other businesses. However, software firms sell software for a wide variety of applications to many different customers.

## POOR SOFTWARE IS VERY COSTLY

Earlier GAO reports and our work on this review have shown that ineffective software can be very troublesome and costly in a variety of ways.

The following findings from our earlier reports illustrate the impact Federal software problems can have.

### Serious impact on user tasks

In 1977 we stated that the Social Security Administration paid recipients an estimated \$1 billion erroneously from January 1974 to December 1975. (See app. II, no. 10.) A large number of these overpayments were attributed to incomplete, untested, or erroneous computer programs. We believe that using the management technique of an independent quality control group, using the structured programming technique to construct the programs, and using tools called program analyzers and test data generators would have improved the programs and reduced their adverse impacts.

### Redundant software efforts

We cited an example where the National Oceanic and Atmospheric Administration spent about \$265,000 in developing its existing personnel system which supported 14,400 employees during fiscal 1975 and 1976. (See app. II, no. 2.) The agency also spent an additional \$470,000 to maintain its payroll and personnel systems. During the same period the Maritime Administration, with approximately 1,500 employees, spent about \$266,000 to develop a new personnel system, while the other operating units in the Department of Commerce collectively spent about \$1 million to maintain their separate payroll and personnel systems. The duplication of effort and the high costs of operating and maintaining these administrative systems--which do very similar tasks--have been identified and reported to management on several occasions. The use of software tools can aid the search for, and evaluation of, software that is suitable for reuse.

### Excessive software conversion costs

We stated that the conversion of software to different hardware systems is a recurring, frequent, and costly activity in the Federal Government. (See app. II, no. 8.) We estimated the annual Federal cost of conversions at over \$450 million, and we estimated that in the then current environment \$100 million could be saved annually, in part by the better use of software tools and techniques.

In our visits to Federal and private ADP installations for the present review, we found similar software problems. Some examples follow.

Excessive costs and frustrations  
from a poorly designed supply system

A software development and maintenance group at a military ADP installation was devoting more than twice as much programmer staff time to maintain an inefficient, obsolete supply system than it was devoting to new development work. Recently, almost all of the group's programmers have worked on maintenance. Even though user requirements and the volume of production have increased greatly, the system has not been improved significantly.

Agency officials referred to the system, which was last redesigned in 1968, as the "original spaghetti program." 1/ According to the system's manager, excessive costs and delays resulted primarily because:

- Most applications programmers did not make much use of modern programming techniques such as structured coding.
- Programmers have been allowed to make user-requested modifications without regard for overall, long term system efficiency.
- There are no specific standards for techniques to maintain the supply system to augment the weak, general installation standards.
- System documentation is poor.

The system manager stated that as a result of these deficiencies, the programs cost too much to operate, are poorly designed and inefficient, and are difficult--if not impossible--to maintain with any degree of reliability.

Capacity of new computers lost  
due to inefficient software

In 1978, a large military data processing center serving 51 organizations purchased a very large computer to satisfy its long term user requirements--some of which had already been delayed in anticipation of a bigger computer. After a

---

1/A term used to refer to programs which are very poorly organized.



costly conversion project made the programs run on the new computer, its capacity was immediately filled. This meant that the present work alone used all the capacity of the new computer, leaving no room for intended expansion, and no capacity to satisfy new user requirements--the original reason for buying the new computer. When we visited the installation, it was making some minimal efforts to promote software efficiency and use software technology. However, no tools had been used to identify inefficient software, and no significant optimization projects had been undertaken to improve the performance of applications software. We believe that performance improvement work would recover a significant portion of the new computer's capacity, allowing it to be used for the new tasks desired by the users.

Ineffective communications system did not meet all user needs and incurred time delays and cost overruns

In the development of software for large, complex communications systems, properly applied software management techniques can ensure adequate management control over the design and documentation. At a large civilian scientific facility, officials told us that they would have saved an estimated \$1,123,000 in programmer labor costs from fiscal 1974 through 1978, if they had spent an additional \$105,000 to document certain software adequately. Currently, the software meets only 80 percent of the user's original performance specifications.

The software that was developed in the final phase of the development project did become operational in mid-1976 but required major maintenance to recode, document, and improve. We were told by facility officials that the unnecessary cost of over \$1,123,000 was incurred because of (1) rewriting unintelligible programs (\$213,000), (2) programmers not familiar with the code (programs) preparing new and better documentation (\$432,000), (3) the difficulty of maintaining and enhancing the system with poor documentation, and (4) lost production runs (\$460,000).

BETTER SOFTWARE TOOLS AND TECHNIQUES WOULD HAVE REDUCED OR AVOIDED THOSE PROBLEMS

Agencies which opt for modern software tools and techniques may help their ADP operations to:

- Reduce adverse impact on user tasks. The modern technique called structured programming produces computer programs which are easier to test, and once tested,

easier to modify. Thus structured programming can both reduce the chances of errors in the user results (for example, overpayments) and make it easier and quicker to respond to future user requests for modifications. Also, appropriate tools can reduce the work of verifying that test data have actually "exercised" a program. This improves the chances of removing errors from the program before it is placed into production.

- Reduce operating costs. Appropriate tools automatically provide information necessary to modify computer programs and make them cost less to run on the computer.
- Reduce overruns and delays. Modern design and development techniques, including structured programming, can make software development more visible to management and more controllable.
- Reduce redundant software projects. Modern software tools and techniques make it more feasible for organizations to reuse existing software and avoid the expense and delay of developing their own software. Tools reduce the labor of analyzing software for suitability, and modern techniques give a better idea of what to analyze for.
- Reduce software conversion costs. As noted in an earlier report (See app. II, no. 8) and expounded by contractors who specialize in conversion, appropriate tools can significantly reduce the labor of making programs written for one brand of computer run on another.

Productivity in software development has shown little improvement over the years

Software has become the dominant ADP cost and is fast becoming one of the most labor-intensive industries in the country. The dramatic improvements seen in hardware reliability, price, and performance have caused hardware to become less important than software because software has not had corresponding improvements. Software costs are reportedly about 70 to 80 percent of total ADP costs and are expected to reach 90 percent by 1985. One industry expert predicted that by 1985, the ADP industry could be the most labor intensive of all industries.

Currently, the leading software development firms are using modern software tools to improve productivity. These

firms' successes overwhelmingly confirm the value of software tools and techniques. Unfortunately, many Federal and private ADP organizations are not following suit.

Government computer applications, which include sophisticated military systems as well as public service systems, often must use complex, state-of-the-art software to handle the huge volumes of data and meet short deadlines. The ability to effectively develop and maintain large volumes of complex software is critical to many Government agencies. The need to increase productivity in these areas is pointed out by a recent NBS publication: (See app. II, no. 33.)

"The preparation and maintenance of computer software is a laborious, human task that can require the coordinated effort of dozens or even hundreds of programmers. The possibilities for problems may be higher than any other technical field because of extreme technical complexity \* \* \*."

We believe that more effective use of software tools and techniques can significantly improve software productivity by reducing the labor for development and maintenance of programs and documentation, and by aiding the reuse of existing programs.

Our questionnaire respondents indicated, and our followup visits verified, noteworthy examples where labor was reduced and productivity improved by software tools and techniques. Some of these are discussed in chapter 3.

## CHAPTER 3

### BETTER SOFTWARE TOOLS AND TECHNIQUES

#### NOW AVAILABLE HAVE IMPROVED

#### MANAGEMENT CONTROL AND SAVED MONEY WHEN USED

Despite the diversity in the use of software tools and techniques discussed in chapter 2, we found that the organizations which use them reap significant benefits. Proper use of software tools and techniques can help

- improve management control;
- improve software quality, which reduces testing and rework, and makes future maintenance, modification and conversion easier;
- reduce the work and cost of software development;
- allow equipment purchases to be deferred because newly written software requires less machine resources to run it, or because existing software can be modified to make it require fewer machine resources;
- reduce costs associated with software during its useful life including the labor costs of maintenance, modification, and conversion, and the cost of the machine resources required to run the software; and
- make the search for suitable software more feasible, thus helping to avoid costly duplicate developments.

Management can promote effective tool and technique use by having quality control groups independent of the software developers and maintainers. Such groups can inspect software and ensure that it complies with quality standards.

We experimented to verify savings through the use of software tools and techniques, and verified the potential for using tools developed by one installation at another.

#### MANAGEMENT CONTROL CAN BE IMPROVED

Projects to develop and maintain computer software can be very lengthy and involve many people. The possibilities for errors in the computer programs and poor management of the efforts are high due to extreme technical complexity and the lack of standards and common practices. By requiring proper use of software tools and techniques, ADP managers and

supervisors can substantially improve control over software development and maintenance.

For example, requiring independent inspection by a software quality control group will help ensure that programming language standards and programming practices standards are followed by the systems analysts and programmers. Appropriate software tools can aid management control by automating much of the manual effort needed to evaluate unfamiliar programs.

Another means of improving management control is to adopt a carefully considered group of software tools and techniques to yield more predictable software costs and provide better documentation. For example, we visited a private sector company which develops software. The company has adopted, and requires the use of, a group of modern programming tools and techniques including structured programming, a program support library, structured design, concurrent documentation, and preprocessors. The benefits reported to us include improved project control, better end products, better organization for the maintenance phase, and estimated annual savings of \$1 million in the development and maintenance of systems.

#### SOFTWARE QUALITY CAN BE IMPROVED

In many systems being developed today, the quality of the software is a significant factor limiting total system effectiveness. Rigorous quality assurance disciplines have been imposed on hardware for many years, resulting in highly reliable hardware. Similar disciplines and practices were not imposed on software, and it frequently lacked the required reliability necessary for a satisfactory product.

Generally, software quality can be improved by applying appropriate tools and techniques in the development phase. For example, programs not sufficiently tested will sooner or later fail because not all errors have been identified. One official believes correcting such failures in large programs may cost up to \$50,000 in computer personnel costs. Software tools can reduce the labor of preparing test data and verifying that the test data has caused the logic of the programs to be exercised. More thorough testing becomes feasible and more reliable programs result.

A management technique to improve the quality of software systems is to require the use of quality control groups independent of the software developers. This quality control can be done by performance evaluation groups or by internal auditors. These groups can review either software development

or maintenance projects. For example, one of our recent reports highlights a case where internal auditors' use of software tools and techniques resulted in the detection and correction of errors prior to system implementation and eliminated unnecessary program instructions. (See app. II, no. 7.)

During our visits to Federal and private organizations, we found other such instances. At one facility, an internal audit group found a software optimization tool which was available for the programmers' use, but was not being used. The audit group used the optimization tool on a large program in the system and found that the storage space needed to run the program could be cut by 36 percent. The audit group recommended that the tool be used more often.

In another case, the same internal audit group found that a software verification and validation tool was not used during acceptance testing of a system being developed by the installation. The audit staff stated it used a tool to analyze three executions of a particular COBOL (common business-oriented language) program on test data and found that many of the program's statements were not exercised at all by the test data. This meant that the test data sets were not sufficient to test all the program's capabilities. The audit staff recommended that the organization use the tool and establish guidelines and standards on using it to test software.

The quality control teams or groups that do exist are usually under the authority of ADP management but often independent of software developers. This type of quality control group is generally charged with monitoring and improving the hardware and software practices within the facility. The improvements resulting from such a group's recommendations can be changes to production software to reduce operating costs. Such groups can test new software before it is approved for production to ensure its reliability and evaluate its efficiency. They also can recommend more efficient and effective procedures for developing, operating, and maintaining software and for using software tools and techniques to accomplish these goals.

COSTS OF DEVELOPING PRODUCTION  
APPLICATIONS SOFTWARE CAN BE REDUCED

The proper use of software tools and techniques can substantially reduce application software development costs because such tools and techniques offer the ability to:

- Deliver new software that is better tested and less likely to contain hidden errors, thus avoiding the costs of errors and later repairs.

- Produce software that is easier and cheaper to maintain and modify during its operational phase.
- Reduce the high labor costs of a highly labor-intensive activity by automating many of the tasks included in software development.

We found several examples of benefits and savings obtained in software development through the proper application of software tools and techniques. On a civilian Federal facility's project, officials showed us their documentation of a 13 percent improvement in programmer productivity due to applying a structured design technique to software being developed. This improvement amounted to an estimated \$300,000 savings over the 25-month development period. For another project, this same facility reduced program development costs primarily by using better software tools and techniques including pre-processors and structured programming. The estimated time to test and install the software built by this project was reduced by 69 percent, and an estimated \$89,000 was saved.

Several Federal and private organizations have reported savings and improved effectiveness in software development as a result of using software tools and techniques in the writing of software. For example, a Federal agency conducted an evaluation of writing and testing computer programs in the conventional method versus using interactive programming. <sup>1/</sup> The results of the test indicated that a given programming task took 96 percent less time using interactive programming than it did with the conventional method.

Savings in the testing of software during the development phase were recorded by a private firm comparing the actual labor costs of five projects being developed using modern programming practices to the labor costs that would have been expected if traditional development practices had been used. Overall, the company reported that project costs were reduced as much as 84 percent on some projects and the greatest part of the savings occurred during the test phase. Also, 45 percent of our respondents reported overall development costs reduced by 11 to 30 percent due to the new software tools and techniques they have adopted in the last 4 years.

---

<sup>1/</sup>Interactive programming provides programmers with the capability of making changes to programs, and testing those changes very quickly. Interactive programming is made possible by software which allows the programmer to communicate with the computer through a quick-response terminal.

## EQUIPMENT PROCUREMENTS CAN BE DEFERRED

We found that the proper application of software tools and techniques to make applications programs run faster and require less computer storage can reduce the computer resources needed to run installations' applications and thus postpone the need to get more expensive, bigger computers. Officials at a military installation estimated that, in the last 4 years, 20 percent of their computer capacity, costing users \$1.08 million per year, has been freed by performance improvement work--commonly called optimization. The tools and techniques predominantly used were fault reporting, 1/ program performance improvement, use of independent quality control groups, and inspections. If performance improvement projects had not freed hardware capacity in recent years, a major purchase of a larger computer would have been necessary. At a civilian installation, \$2.4 million in personnel and computer resources was freed by using tools and techniques including program analyzers, program optimization, an independent quality control group, and fault reporting. An agency official stated that the various optimization projects at the center helped the center delay an equipment acquisition.

A noted software performance evaluation expert cited an example where performance improvement of a securities transfer system allowed a private firm to cancel an order for over \$2.5 million worth of new equipment. This firm was a major utility company with two large computers, one dedicated entirely to a stock transfer program utilizing 60 percent of its capacity. Since an anticipated stock split within 6 to 7 months would have caused the computer workload to double, the company decided to buy larger computers to meet the increase. After the purchase proposal was drafted, external consultants reviewed the stock transfer system, pinpointed problems with its edit checks, and recommended improvements. The recommendations saved 25 to 30 percent of the computer time consumed by that application and allowed the intended purchase to be canceled.

At another agency we visited, the computer performance evaluation team and the internal review staff made several changes which improved the efficiency of production software. The value of resources freed over the 1976-78 period was estimated at \$470,000. In the past few years at a military installation, the performance evaluation group dedicated to developing software monitoring tools, optimizing ADP systems,

---

1/Fault reporting is a formal process for documenting observed errors.



and promoting efficient programming practices saved over \$130,000 worth of freed resources in yearly production costs. These savings are due to applications program performance improvement which reduced the machine resources required to run applications software, freeing the resources for other work.

COST OF MAINTAINING, MODIFYING,  
CONVERTING, AND OPERATING PRODUCTION  
APPLICATIONS SOFTWARE CAN BE REDUCED

Production applications software generally incurs costs in three main areas during its useful life: maintenance costs, modification costs, and operating costs. Such software also may incur conversion costs. In the first and second areas, the costs consist mainly of labor and computer time used to make and test changes to the programs. In the third area, the costs consist of computer time and supplies such as paper. The fourth cost area--conversion--happens when the software must be made to run on a different computer than it was written for. By requiring that first line supervisors control these areas, management can significantly reduce costs.

Software tools can reduce all four categories of costs. Some tools can reduce the labor cost of making and testing changes to programs--whether they are made for modification or for conversion--and other tools can aid in modifying programs so they will take less computer time (called performance improvement or optimization).

Software tools and techniques can reduce the labor costs of maintenance. A recent issue of a technical magazine stated that maintenance costs can more than equal or double the original development costs, assuming at least a 5-year operating life for the software. We found several examples where the use of tools and techniques reduced these costs. At a military installation, officials reported an estimated \$5 million savings in maintenance and modification costs over 4 years through the use of a combination of software tools and techniques, which included independent quality control and tools for evaluating program performance.

The proper application of software tools and techniques can reduce costs of the other part of the operational phase, modification. Several agency and private sector ADP officials and software authorities stated that the time required to modify software can be cut by as much as 50 percent with modern methodologies including structured programming, top-down program design, and configuration management. At one military installation, officials reported that the use of software tools and techniques including structured programming, text

editors, 1/ program flow analyzers, and formal inspections has saved them over \$126,000 per year in maintenance and modification expenses. The impact of software tools and techniques was further supported by responses to our questionnaire. Over 45 percent of the respondents reported reduced overall maintenance costs from 11 to 30 percent due to new technology they adopted in the last 4 years.

At a civilian agency with a software production management and control group, officials estimated savings at \$50,000 per year with configuration management methods used to control software modifications. Configuration management involves close control and documentation of modifications to operational software to ensure that continuing service to the end users is not adversely affected by the changes, and to ensure that modifications are justified to, and approved by, management before they are made. These officials claimed that the use of configuration management methods has greatly aided management in controlling software and has increased the productivity of the programmers.

Software conversion costs are incurred in several ways, including delays in making user tasks operational on replacement computers, extra labor, retraining, and documentation. (See app. II, no. 8.) Software tools and techniques can significantly lower such conversion costs. Tools and techniques can be applied during the original development of software to add quality and ease conversion. The means of attaining higher quality include better enforcement of standards (for programming languages, programming practices, and documentation), adoption of new programming technologies, and better training for programmers and analysts. Tools and techniques used during conversion to reduce time and labor include automatic translation programs and text editors.

We visited a private firm that used several software tools in a conversion, including a language translator, a pre-processor and a text editor. The firm estimated these tools saved \$97,000 and avoided substantial conversion delays. The firm's computer vendor had estimated, based on experience, the conversion would require about 5 staff years, but the firm made the conversion in 4 months, using only about 7 staff months. Also, military officials estimated a savings of about \$232,000 in programmer labor costs from using a COBOL conversion tool and a text editor for converting the installation's COBOL programs to a new system.

---

1/A text editor is a computer program used to erase, insert, change, and move words or groups of words. Those words can be part of a computer program or ordinary English text.

We found several installations which require applications programmers to use software tools to supplement the use of standard higher level languages and reduce operating costs. One private sector facility routinely requires a program analyzer 1/ to be used on all programs before they are released into production. The company estimated that \$508,700 has been saved in computer usage time over the 7-year period the tool has been used. Another firm estimates benefits in excess of \$85,000 annually from adopting similar methods. This firm believes that providing such tools to supplement applications programmers improves the efficiency of coding technique and improves productivity by decreasing debugging and testing time. In both cases management felt that requiring programmers to use available software tools improves management control of software development and maintenance.

At a military installation, we found that the operating efficiency of one applications program was improved 98 percent with program analyzers. These tools identified the parts of the program requiring the most time to run on the computer (execution time). Programmers were able to improve them so they would run in much less time. The estimated annual savings attributed to this applications improvement was over \$34,200.

We found that programs can often be modified to reduce their machine operating costs without making them more difficult to maintain or convert. In other situations, vendor-unique features (features peculiar to one brand) may offer such large operating costs savings on the present computer that the difficulty their use adds to later conversion is justified by the immediate savings. However, we believe that use of vendor-unique features should be restricted by management to those situations where savings due to their use are clearly provable 2/ and that such use should be well documented so that later conversions to replacement systems will not be impeded. Generally, we believe that any changes made to programs to reduce their machine costs should be carefully evaluated concerning the impact on other software costs such as the cost of incorporating functional changes that users might request later.

PRODUCTIVITY CAN BE RAISED  
BY AVOIDING REDUNDANT DEVELOPMENTS

We feel that much software effort is wasted (with a consequent loss of productivity) in developing redundant

1/Program analyzers are defined on page 5.

2/For example, in large programs which are frequently used several years.

software--writing new computer programs and documentation for tasks which have been programmed many times. The private sector has dramatically increased its use of readymade computer programs for common applications such as payroll, but the overwhelming tendency in Federal agencies is to make, or contract for the making of, new software.

The readymade programs used in the private sector are primarily software products--programs offered by software firms. Government agencies can share programs that are already Government property, as well as acquire products. Existing programs that can be shared or bought to avoid the cost and delay of new development include systems programs, utility programs (including software tools), and applications programs.

In the past, Federal organizations resisted using programs developed elsewhere, claiming that the programs did not meet the organizations' unique needs, that it would require too much effort to convert the programs to run on the organizations' computers, that no maintenance was available from the developers, and that the programs were poorly documented. Yet, despite these claims, we know of a few instances where as much as three-fourths of the cost of new development was saved by Federal agencies that adapted existing applications programs to their needs.

Modern software tools make a search for suitable applications programs more feasible because appropriate tools can automate much of the labor of analyzing and inspecting programs proposed for reuse so that their suitability can be assessed much more quickly. Thus, the traditional objections to reusing programs are weakened.

We found a significant example where a military installation developed software itself only to later scrap it and buy commercial software. Four applications in a financial system (which cost about \$1,637,000 per year to operate) were to be redesigned and linked to a central data base through a data base management system (DBMS). 1/

Despite a preliminary finding that two commercial DBMS's were suitable, the agency decided to build its own by augmenting the existing programs. The argument given was urgent need, and the time and cost were estimated at 1 year and \$320,000. After about 2 years, the system manager estimated

---

1/A data base management system is a set of systems programs which can facilitate the management, manipulation, and control of data.

that the original estimates would be exceeded by 4 years and \$710,000. We were also told that, even when completed, the in-house DBMS would not satisfy some important user requirements.

Finally the agency decided to get a commercial DBMS after all. This will not only waste what has been spent already, but also will require two added costs: an estimated \$906,000 to modify the applications again for the new DBMS, and the cost of the commercial DBMS itself. <sup>1/</sup> This example suggests that complex systems programs such as DBMS's should be acquired readymade from professional software developers, not developed in-house by users.

In summary, management should more strongly emphasize searching for existing software--either commercial or Government owned--before developing new software. Reusing software may be considered a modern software management technique in that there is now an inventory of existing software that may be reused which did not exist before. Software should be inspected and evaluated carefully before the decision is made to reuse it--whether it is for sale or free from other agencies or sharing centers. Some software tools can be used to aid the inspection and evaluation and make reuse of software more feasible.

#### OUR EXPERIMENTS VERIFIED THE POTENTIAL OF REDUCING SOFTWARE COSTS

As part of this review, we tested both the techniques of improving the performance of applications programs and the possibilities of using tools developed at one installation at another. We found several locally developed software tools and methods for improving performance of production programs at one site. In order to effectively apply these tools and evaluate their benefits, we worked with an installation staff member to improve the performance of two of the installation's production programs. The projected first-year savings in operating costs from using the improved programs totaled about \$9,000.

We then took one of the tools which had effectively been used at this installation to another Federal installation. We used the tool and selected program changes to improve a production program, and reduced its execution time by 71 percent. The projected first-year savings in operating costs from using the improved program were about \$25,000.

---

<sup>1/</sup>Commercial DBMS software typically costs from \$40,000 to \$120,000 depending on the particular DBMS and the options acquired.

The basic methodology was to identify programs with high operating costs, then select one and apply the software tool to get information on how the program worked. Improvements were made to parts of the program which repeated many times to make them go faster--while still meeting the same user requirements. (The tool counts the number of times each part of a COBOL program is executed. This processing activity count aids analysts in finding the parts of the program that are likely candidates for improvement.)

All three programs we selected for this experiment were real applications programs. We, and the agency personnel, considered them typical. They represent what can happen to computer programs in production: unless changes made to them are monitored by some form of performance evaluation group, the programs can eventually become quite costly to operate. According to several software authorities (and supported by our own results) many installations have a few of the programs which consume most of the computer resources--one-fifth of the programs consuming about three-fourths of the resources is not uncommon. Therefore, performance improvement of the expensive programs is quite worthwhile.

The projected first-year savings due to our experiment totaled about \$34,000 at the two installations. We spent about 4 staff weeks on the experiments. We also demonstrated that software tools developed at one installation can be used at another with high potential for cost savings--avoiding duplicate development of tools--and with limited resources, and that the techniques which the tools aid can be applied at many installations.

## CHAPTER 4

### WHY MODERN SOFTWARE TOOLS AND TECHNIQUES

#### HAVE NOT BEEN MORE WIDELY USED IN THE GOVERNMENT

We believe there are several factors contributing to the lack of widespread use of modern software tools and techniques in the Government, including:

- Lack of Government-wide policy, guidance, and standards. OMB, GSA, and NBS directives do little to aid or direct agencies' management to seek better ways of using software tools and techniques in designing, developing, testing, and maintaining computer software.
- Failure of ADP management to require that in-house and contractor developed software be inspected regularly. Such a requirement would increase the use of the types of tools which aid inspection.
- Lack of emphasis on software improvement by agency or installation managers or by independent review groups. This has caused the use of new software tools and techniques at some installations to be left up to the interest and motivation of individuals.
- Lack of effective, coordinated, Government-wide efforts for the sharing of tools and for the research and development of better tools.

We identified these problem areas through a combination of discussions with OMB, GSA, and NBS officials, our mailed questionnaires (which drew 515 responses from Government officials), visits to agencies to verify questionnaire responses, interviews with experts, and a review of current literature.

#### LACK OF GOVERNMENT-WIDE POLICY, GUIDANCE, AND STANDARDS

Current Government-wide ADP policy, guidance, and standards do not specifically address development, use, and evaluation of software tools and techniques. In general, we found little or no OMB, GSA, and NBS guidance for managers on the specific actions they should take to improve their software. Also we found no procedure for checking software for compliance with standards. For example, a recent GAO report states that agencies have not followed the one existing Federal programming language standard (COBOL). (See app. II, no. 3.)

OMB policies and GSA and NBS regulations give very little guidance to agency heads and ADP managers on using software

tools and techniques. OMB Circular A-54, August 26, 1971, addresses the policies on selection and acquisition of ADP equipment. This circular does not mention specific performance improvement techniques to relieve workload saturation on an existing computer. The regulation is predominantly hardware oriented.

OMB Circular A-71, March 6, 1965, addresses the responsibilities for the administration and management of ADP activities and states that the heads of executive agencies are responsible for developing data systems using the most advanced design techniques. However, the circular does not tell ADP managers how to select and apply those advanced design techniques. (Transmittal Memorandum no. 1 to OMB Circular A-71, July 27, 1978--the latest memorandum--is concerned with security.) A November 13, 1978 draft revision to A-71, was circulated for comment. It cites the great importance of information technology and says that heads of departments and agencies have primary responsibility for using it efficiently and effectively. However, the draft makes no specific references to new software technology.

OMB Circular A-109, April 5, 1976, provides vague and limited guidance on the development, use, and evaluation of software as related to the acquisition of a major system. OMB Circular A-113, November 17, 1976, required ADP management to address ADP inefficiency problems in management plans; however, this circular was suspended 4 months later. OMB officials stated they have published very little policy on the acquisition, development, use, and evaluation of software. In our opinion, OMB regulations lack the guidance and direction ADP managers need in using software tools and techniques.

Many of the Federal installation ADP managers and supervisors we contacted reported a need for more specific guidance in using software tools and techniques. In fact, over 80 percent of the Federal installations responding to our questionnaire indicated a need for Government-wide directions and standards for development, sharing, and use of software tools. At most Federal installations we visited, the available organizational guidance for the acquisition, use, or evaluation of software tools and techniques was limited or nonexistent. At several sites where higher level agency guidance or procedures did exist, they were of a general nature and usually ambiguous--and did not provide specific instructions on why and when the various tools and techniques should be used. We discussed this lack of guidance with GSA officials; they agreed that they had provided very little formal guidance to installation managers or operational level supervisors concerning better ways of using software tools and techniques to improve ADP operations.



Recently NBS AND GSA issued some helpful but limited guidance. For example, in November 1978, GSA published "Management Guidance for Developing and Installing an ADP Performance Management Program." This comprehensive publication suggests various ways ADP facility managers can establish and develop an internal performance evaluation program to review the efficiency and effectiveness of their day-to-day operations.

GSA's Federal Property Management Regulation (101-35.206(a)(3)), states that, in determining the need for acquiring more ADP, agencies should determine the possibility of improving the performance of existing data processing facilities. However, the regulation does not mention how to improve performance, nor does it mention improved software technology as such. Our suggested augmentation to the regulation is in appendix I.

NBS has, within the last several years, issued five documents on the use of software tools and techniques. (See app. II, nos. 33, 34, 36, 38, and 39.) Just recently, NBS initiated a project to design and develop a software cost estimation model for Government use. We discussed the lack of standards and guidance in the development, use, and evaluation of software tools and techniques with NBS officials. They agreed that such guidance, especially standards, is badly needed by the ADP managers and supervisors; however, they stated that funding constraints have curtailed some of their plans in this area. The need for guidance is further demonstrated by our questionnaire: over 68 percent of the Federal agency respondents reported that they have no formal rules or standards for evaluation of the effectiveness of software tools and techniques after use. Also, over 40 percent of these respondents reported that they have no formal rules or standards for the use of software tools and techniques.

#### AGENCY MANAGERS OFTEN FAIL TO REQUIRE INSPECTION OF CONTRACTOR-DEVELOPED SOFTWARE

Most Federal agencies have failed to inspect contractor-developed software to the extent made feasible by modern software tools. We believe substantial Government-wide benefits could be attained through this inspection.

Several private sector software firms we visited have used software tools and techniques to monitor and inspect software developed by subcontractors. Software which did not meet the companies' requirements was returned to the subcontractor and was not paid for until it did meet requirements. Several companies reported significant savings as a result of

inspecting contractor-developed software. For example, one firm's officials said using automated aids to verify adherence to contract standards helped save over \$100,000 on a software project. Officials at another private facility attributed numerous benefits to using software tools and techniques.

NEITHER MANAGEMENT NOR INDEPENDENT  
REVIEW GROUPS EMPHASIZE SOFTWARE  
IMPROVEMENT

At most of the Federal installations visited, we found management was unaware of how independent review groups, using software tools and techniques, can (1) review software practices, (2) improve use of computer resources, and (3) improve the design and development of systems. As a result, there are few independent review groups, and the benefits they could provide are lost.

A 1978 report from the President's Reorganization Project for Federal Data Processing supports the view that such groups are beneficial, but are not generally in use. (See app. II, no. 25.) The report states that:

- Systems assurance activity in the systems development process is the exception rather than the rule. Meaningful ADP audit or evaluation involvement in the systems development function is nonexistent or inadequate in many agencies.
- When ADP audits have been instituted and conducted properly, they have been very effective in detecting inefficient practices, improper organizational alignment, and improperly trained personnel. Most agencies do not have enough trained ADP auditors to do such reviews as part of their normal audit activities.

We found that many internal audit groups do not review software applications or software design, development, and maintenance practices. At some Federal installations visited, the internal audit staffs lacked the ADP skill and knowledge necessary to perform such reviews, or confined themselves to financial ADP systems. The President's reorganization project report mentioned earlier and a previous GAO report indicate that the lack of technical ADP knowledge by auditors hinders their performing effective ADP audits of system design and development, specific applications, and ADP management practices. (See app. II, no. 7.)

In March 1979 GAO published "Auditing Computer-Based Systems," <sup>1/</sup> a supplement to its 1972 audit standards publication. The supplement addresses the involvement of internal audit groups in the design and development of systems as well as in review of controls in computer-based systems.

The use of a computer performance evaluation group and/or a quality assurance group to monitor software development and maintenance was uncommon at most Federal installations visited. There is a lack of management support for the use of independent review groups to improve the design and operation of software systems. In September 1976, the House Committee on Government Operations reported a lack of management attention to achieve economies in the use of ADP and concluded that agencies should strive to optimize their ADP to achieve highly economical and efficient programs and avoid improper design, inefficient applications, or operational deficiencies. The President's reorganization project reports indicate that agencies are still lacking in the above areas. For example:

- Quality assurance (control) in the systems development process is the exception rather than the rule.
- There is a basic lack of awareness in many installations of the effect detailed software procedures can have on overall systems performance.
- There are numerous examples of earlier generation software still being maintained and operated on newer systems for which it is not well suited.

OTHER REASONS SOME INSTALLATIONS USE  
NEW TOOLS AND TECHNIQUES SO LITTLE

At many of the 33 Federal installations we visited, the programmers and supervisors who were trained in new software technology were not using or applying what they learned. Their reasons included:

- Management provided no support or incentives for using new software technology.
- New software technology is perceived as not required for new software development.

---

<sup>1/</sup>This new publication (stock no. 020-000-00174-7) may be obtained from the Superintendent of Documents, U.S. Government Printing Office, Washington, D. C. 20402.

--Use of new technology is limited to individual technical competence and interest rather than deliberate management requirements.

Our interviews with agency personnel and our examination of sample programs and programming standards manuals supported the reasons cited above. For example, at several installations where officials stated structured programming was used, we examined samples of the production applications programs. We usually found the samples were not written using structured programming. In other instances, programmers stated that they had been trained in structured programming, but that their managers did not support its use.

At several of the Federal facilities visited, we found instances where using new software technology was not required for new software development even though it was available. Also, we found several instances where the use of new technology depended on individual initiative and technical competence rather than on a systematic organizational effort. Several supervisors had their own interpretation of structured programming. For example, at one naval facility we visited, the application of structured programming varied substantially among its five different software-producing groups.

Recent reports from the President's reorganization project support our findings cited above. For example:

- The summary of the Human Resources Team Report states that the proper environmental incentives and motivations for the innovative use of information technology do not exist within the human resource agencies. (See app. II, no. 26.)
- The summary of the Small Users Team Report states that data processing performance leaves much to be desired, with agency top management primarily responsible for the condition. (See app. II, no. 27.) Agency management has generally not provided the necessary support, attention, direction, or interest to the proper application of information technology.
- The Operational Management Team Report states that many agencies have not developed or adopted a formal approach for the development of systems, project management, or the statement of specifications and procedures for contractor performance. (See app. II, no. 25.) This permits an undisciplined development of systems, often by personnel with marginal systems skills. The end result is too often a systems product

which does not meet user requirements. This leads to additional costs during production for "maintenance and enhancements" to patch and repair the original system. Often, those agencies that do have documented development methodologies fail to follow them.

#### POOR SHARING OF TOOLS WITHIN AND BETWEEN AGENCIES

Now there is no effective formal mechanism for sharing software tools in the Federal Government either within agencies or between agencies. Thus (1) the benefits from the use of software tools are not widely available and (2) installations which develop tools locally often duplicate the others' work. Even though participation in the Federal Software Exchange Program is mandatory for Federal agencies according to Federal Property Management Regulation 101-36.16, fewer than 4 percent of our Federal respondents reported that they have obtained shared software from GSA's Federal Software Exchange. Further, 63 percent perceived a need for a Government-wide clearinghouse for both software tools and techniques.

A recent GAO report on software sharing and several of the President's reorganization project reports support the points just cited. For example:

- The GAO report cites that the Federal Software Exchange Program, in its present state, is only a small step toward achieving more computer program sharing among Federal agencies. (See app. II, no. 6.) The program has had very little use to date.
- The President's Reorganization Project Standards Study Team final report, June 1978, states "the level of software sharing in the Federal Government is minimal, if existing at all." (See app. II, no. 24.)
- The President's Reorganization Project Small Agencies Team draft, May 1978, stated that most agencies visited were reluctant to use the GSA Software Exchange Program; some had not even heard of it. (See app. II, no. 22.) A common complaint was difficulty in finding software that would meet particular needs. Also, there was reluctance to contribute software to the program, and those who did feared they would have to aid another agency with the adaptation of the software or with subsequent maintenance.

A major argument of officials at several Federal installations for not participating in the sharing program--an argument also noted in the President's reorganization panel

reports--is that the added cost of making new software suitable for sharing with other installations is never paid back to the original developers. The officials stated that software developed for sharing (either software tools or portable applications software) requires an established maintenance function to support it after it has been delivered to other installations. For example, if a text editor is shared with 15 other installations, they should be notified of upgrades and/or modifications to it and should receive copies of the new version from the original developer. However, this maintenance function would detract from other work at the facility which developed the software, especially if that facility were not paid for the maintenance work.

We visited several Federal facilities that were benefiting from the development of portable software and software tools. Officials at one facility estimated that \$6 million could be saved in future conversion costs as a result of the management actions and procedures they had developed to design their software for portability. Furthermore, several software development tools have been easily converted to different types of hardware, allowing the designers and developers to utilize these tools in developing software on other hardware. Several of the private firms visited supported the views expressed above.

Perhaps the most common method of sharing software tools among Federal agencies has been accomplished through informal software exchanges and in groups using only a particular vendor's computers such as SHARE 1/ or USE. 2/ However, spokespersons at the Federal and private facilities we visited indicated that programs obtained from such groups are utility type programs usually written in assembly language, and therefore usable only on a particular vendor's computers.

#### LACK OF COORDINATED RESEARCH AND DEVELOPMENT

Presently, there is a lack of coordinated research and development in modern programming practices including such areas as software tools and techniques. For example, the benefits of DOD's research and development studies on modern programming practices are limited to selected DOD facilities, and to the private contractors who helped develop the tools and techniques. We found several Federal facilities (mostly non-DOD) unaware of, or not sharing, the information produced by the DOD work.

---

1/An IBM user group.

2/A Univac user group.

The Presidential Reorganization Project's Central Agencies Team stated in its draft report, July 1978, "that the non-defense sector of the Federal Government does not have an adequate research and development effort in information technology." Even so, we found that the DOD and sparse non-DOD agency research on modern programming practices and new software technology has resulted in some duplication. To illustrate:

--Recently, NBS developed a software cost estimation model through a private contractor. DOD has been doing similar work, and the Nuclear Regulatory Commission also has completed some research in the area.

--One DOD software research and development center has published at least 275 reports on various software technology areas. We found several non-DOD studies which duplicated the work. For example, NASA funded a three-volume study, "A Methodology for Producing Reliable Software," published in 1976. The DOD software research and development center has published over five reports on this subject since 1974.

At several Federal and private facilities visited, many ADP supervisors and programmers were not aware of (1) the current NBS software publications or (2) the numerous published reports (both Government and private sector) on the benefits of modern programming practices. Several agency officials suggested summarizing such information and distributing it to all agencies. For example, Federal Computer Performance Evaluation and Simulation Center reports could be summarized and shared with other Federal users having the same type of hardware. The Government must coordinate its research and development efforts in modern programming practices. Lack of coordination precludes potential Government users from sharing of useful information.

## CHAPTER 5

### CONCLUSIONS, RECOMMENDATIONS, AND AGENCY COMMENTS

#### CONCLUSIONS

We found many opportunities for greater use of software tools and techniques in many Federal agencies. The use of such tools and techniques at each agency should be based on a careful study of the cost benefits involved--both tangible and intangible. We believe Federal ADP managers must support more actively the implementation and use of new software technology in their operations.

At various Federal agencies we found examples of computer software which cost more than necessary to develop and operate. This has been true for both in-house-developed and contractor-developed software. We found inefficient, ineffective, and costly Federal computer software as cited in earlier GAO reports. In our opinion, developmental and operational costs of Federal computer software will keep increasing if its management, production, and maintenance practices continue in the traditional way.

We found that the Federal use of modern software tools and techniques has, with rare exceptions,

- had no deliberate management emphasis or direction within the agencies or across agency lines;
- been adopted--if at all--in the form of written standards with no provision for inspecting software for compliance;
- had no Government-wide coordination to reduce redundant developments and transmit lessons learned from one agency to another;
- been due to the interest of individuals, not due to deliberate management efforts.

Several things must be done to improve the situation. A Federal Procurement Management Regulation should be augmented to guide agency actions for efficient use of computing resources in more detail. The central ADP agencies--OMB, NBS, and GSA--should act, as recommended below, and heads of Federal agencies should promote using modern software tools and techniques within their agencies.



Where appropriate, coordinated software policies and procedures should be developed and implemented, and organizational structures should be instituted to correct the above-mentioned weaknesses.

We believe that modern software tools and techniques can offer the Federal Government

- better management control of computer software development, operation, maintenance, and conversion;
- lower costs for computer software development, operation, maintenance, and conversion;
- feasible means of inspecting both contractor-developed and in-house-developed computer software for such quality indications as conformance to standards and thoroughness of testing.

### RECOMMENDATIONS

We recommend that the Director, OMB:

Clearly define the responsibilities of agency heads and ADP managers for the acquisition and management of Federal information technology. We believe the recent draft revision to OMB Circular A-71, November 13, 1978, is a step in the right direction; however, to promote efficient and effective software use in the routine Federal ADP operations, we recommend that more specific guidance be given on new software technology such as structured design, program design languages, preprocessors, configuration management, concurrent documentation, and performance evaluation programs.

- Require that all Federal agency heads address the software quality assurance function in their agencies, document such consideration, and where the cost is justified by the potential benefits, establish an ongoing software quality assurance function. This function should be independent of software developers and could be implemented in an agency's internal audit organization, as discussed in GSA's publication "Management Guidance for Developing and Installing an ADP Performance Management Program." The quality assurance group should audit performance improvement of applications software, audit conformance to language standards, and audit conformance to programming practices standards. The group could also be concerned with quality assurance of contractor-developed software. In establishing a

software quality assurance function, heads of Federal agencies should promote modern software practices for software that is developed for their agencies either in-house or by contractors. Such promotion should include agencywide software policies and procedures for adherence to language standards, portable software, structured programming, use of existing software, and inspection for compliance.

- Direct the establishment of coordinated Government-wide research and development for software tools and techniques which will include provision for dissemination of information to all potential Federal users. This should include proper documentation and maintenance for portability of the tools or techniques. Such recommendations are implied in the draft OMB Circular A-71, November 13, 1978; however, we believe more specific reference to software tools and techniques and modern programming practices should be incorporated into the revised circular.

We recommend that the Administrator of General Services:

- Augment paragraph (3) of Federal Property Management Regulation 101-35.206(a) to incorporate what agencies should do to improve their applications software and carry out the intent of the paragraph. Appendix I of this report suggests such an augmentation. GSA and NBS should collaborate on this addition because our suggested revision has both procurement and technical content.
- Establish, by development or adoption, a set of standard tools and methods to solve operational problems, promote efficiency and economy, and inspect software (see below). These standard tools and methods could be used as a starting point for the inclusion of standard software tools in GSA's Software Exchange Center. GSA should use recent NBS publications as guides and collaborate with NBS in establishing the set of tools. (See app. II, nos. 33, 34, and 38.) Several tools exist that might serve the purpose--thus neither NBS nor GSA need necessarily develop new tools. The tools adopted should themselves be written entirely in higher-level languages, where possible, to maximize their portability to different brands of computers.
- Require that contractor-developed software pass a standard inspection, with software tools. The inspection could include a demonstration that the logic of the program was adequately exercised by testing. Such tools and inspections could also be applied to software developed in-house.

--Establish a software tools category at the Federal Software Exchange Center and provide the technical support necessary for sharing the tools.

We recommend that the Secretary of Commerce, through NBS, develop or adopt standards or guidelines (1) for using software tools and methods to promote efficiency and economy and (2) for software inspection. GSA and NBS should collaborate on this since it has both procurement and technical content.

#### AGENCY COMMENTS

We asked the General Services Administration, the Office of Management and Budget, and the Department of Commerce to comment on our draft report. Their replies are included as appendix III to this report and are discussed below.

#### GENERAL SERVICES ADMINISTRATION

The Administrator of General Services agreed with the report's findings and stated that his agency will implement the report's findings to the maximum extent possible.

Concerning our recommendation that GSA augment the Federal Property Management Regulation as suggested in appendix I, the Administrator said that he agrees in principle, but reserves judgement as to the specific wording and location of the change. We offered appendix I to illustrate the type of change we propose, not the final wording.

The Administrator supported our recommendation that GSA establish a set of standard tools to solve operational problems and promote efficiency and economy, and said that GSA agreed with NBS that a list of abstracts on software tools collected by NBS will be added to the GSA Federal Software Exchange catalog. This is a step in the right direction.

The Administrator also supported our recommendation that GSA require that certain standard inspections, using software tools, be done on contractor-developed software. GSA will address the issue in its response to the GAO report on software contracting. 1/

Concerning our recommendation that GSA establish a software tools category at the Federal Software Exchange Center and provide the technical support necessary for sharing the tools, the Administrator said that such activities are

---

1/Published November 1979.

potential functions of the new Software Development Office. 1/ He again mentioned that a software tools category will be included in the next Federal Software Exchange catalog.

Concerning our combined recommendation that NBS develop or adopt software inspection tools and GSA require their use, the GSA Administrator said that this matter also would be addressed in the GSA response to the GAO contracting report.

#### OFFICE OF MANAGEMENT AND BUDGET

The Office of Management and Budget agreed with us that agencies are not always using the latest software technology. OMB went on to say that if any new technology can be cost-justified, OMB would support agency budget requests for that technology. We agree with OMB that proposed new technology should be able to repay its costs.

Concerning our recommendation that OMB more clearly define the responsibilities of agency heads and ADP managers, OMB officials said that they felt that their Circular A-71 generally covers these matters. They said that in light of our findings they will examine GSA's and Commerce's activities in this area and then determine if Circular A-71 should be modified.

#### DEPARTMENT OF COMMERCE

The Assistant Secretary of Commerce commented formally in a letter which included the comments of the National Bureau of Standards. He concurred with the intent of the report but said that the recommendation for NBS action seemed ambiguous or inconsistent. The letter both discussed the recommendations specifically and offered suggestions "to correct or clarify other sections of the report."

The Commerce letter began by pointing out that page vi of our draft said "develop standards or guidelines," whereas page 65 said "develop or adopt tools," and then asked for clarification. Our intent was "standards or guidelines" and we have changed our wording accordingly and clarified the matter of collaboration between GSA and NBS (pages iii, 38, and 39 of this final report).

The letter then informed us that NBS is developing a program analyzer for FORTRAN 2/ programs--the first standard

---

1/An office recently established in GSA.

2/A computer programming language

tool developed under the NBS software quality task. The letter pointed out that NBS does not now have funding or staff to do more than demonstrate the feasibility of such tools through prototype efforts. We realize that tool development is expensive and that NBS may not be able to develop tools on a large scale. We also are aware that numerous tools exist (hence our recommendation that guidelines be developed) and believe that soon considerable gains can be made by better use of existing tools.

The Commerce letter then told us that NBS began several related activities as part of its strengthened ADP Standards Development Program:

- Concerning our discussion in chapter 4 of the need for standards for evaluation and use of software tools, NBS is developing a classification system for tools and for definitions of various programming environments that determine the types of tools needed.
- Concerning our recommendation that OMB direct the establishment of a Government-wide research effort including dissemination of information, NBS has collected information on over 600 software tools and is placing it in a data base system to make it available to Federal agencies.
- Concerning our recommendation to GSA that standard inspections be required for contractor-developed software, NBS is developing guidelines for validation and verification of software for the various Federal ADP environments.

Several detailed comments were then offered "to correct or clarify other sections of the report"; since they are detailed and technical in nature, we discuss them in appendix IV. We incorporated NBS' suggestions where we felt they were appropriate.

## CHAPTER 6

### SCOPE OF REVIEW

We conducted a nationwide review which included a questionnaire to Federal installations and private firms, visits to the three agencies with Government-wide ADP responsibilities, visits to Federal and private research and development activities, visits to Federal and private ADP installations to verify the questionnaire, examination of sample programs, and study of relevant literature. Also, we obtained experts' opinions on modern programming practices.

During our review we summarized the responses to 515 Federal and 408 private sector questionnaires, primarily to determine how software tools and techniques are used in both sectors. We selected and visited several of these facilities and obtained, developed, and documented cost-benefit examples. Altogether, we visited 33 Federal and 25 private industry computer installations, located in the Washington, D.C., Norfolk, Chicago, Kansas City, Seattle, and Los Angeles areas.

During this review, we were primarily concerned with the application of better tools and techniques to applications software on general-purpose computers. As mentioned before, another GAO review is examining the DOD effort to apply better tools and techniques to weapons system software.

SUGGESTED AUGMENTATION OF FEDERAL  
PROPERTY MANAGEMENT REGULATIONS

Subpart 101-35.2 Management, Acquisition, and  
Utilization of Automatic Data  
Processing (ADP)

S101-35.206 Policies and procedures 1/

- (a) Documenting the determination of need for acquiring ADP.
- (2) Workload and data processing requirements have been revalidated by an independent review group to determine if a reduction of nonmission-type work can be effected.
- (3) Action has been taken to determine the possibility of improving the performance of existing data processing facilities through establishment of an agency group for performance evaluation and software performance improvement <sup>2/</sup> and through new technology, including
- computer performance evaluation;  
optimization of existing software;  
use of improved software methods  
such as structured design, pro-  
gram performance evaluation,  
configuration management, and  
quality assurance inspections;
- or interim upgrade or system modifications, rescheduling, software changes, improved work center procedures, or extended shift operations; and \* \* \*

1/Underlined statements are GAO-suggested revisions.

2/As suggested in GSA's publication "Management Guidance For Developing and Installing an ADP Performance Management Program," November 1978.

LIST OF SOFTWARE-RELATED  
GOVERNMENT PUBLICATIONS

U.S. GENERAL ACCOUNTING OFFICE

1. "The Department of the Interior's Computerized Resources Information Bank," EMD-78-17, July 17, 1978.
2. "Inadequacies in Data Processing Planning in the Department of Commerce," FGMSD-78-37, May 1, 1978.
3. "The Federal Information Processing Standards Program: Many Potential Benefits, Little Progress, and Many Problems," FGMSD-78-23, Apr. 19, 1978.
4. "Shifting the Government's Automatic Data Processing Requirements to the Private Sector: Further Study and Better Guidance Needed," FGMSD-78-22, Apr. 11, 1978.
5. "Accounting For Automatic Data Processing Costs Needs Improvement," FGMSD-78-14, Feb. 7, 1978.
6. "The Federal Software Exchange Program--A Small Step in Improving Computer Program Sharing," FGMSD-78-11, Jan. 13, 1978.
7. "Computer Auditing in the Executive Departments: Not Enough is Being Done," FGMSD-77-82, Sept. 28, 1977.
8. "Millions in Savings Possible in Converting Programs from One Computer to Another," FGMSD-77-34, Sept. 15, 1977.
9. "Need To Apply Adequate Controls in the Army Standard Payroll System Prior to Implementation Defense-Wide," FGMSD-77-4, July 5, 1977.
10. "Problems Found with Government Acquisition and Use of Computers from November 1965 to December 1976," FGMSD-77-14, Mar. 15, 1977.
11. "Ways to Improve Management of Federally Funded Computerized Models," LCD-75-111, Aug. 23, 1976.
12. "Problems in Developing the Advanced Logistics System," LCD-75-101, June 17, 1976.
13. "Improvements Needed in Managing Automated Decision-making by Computers Throughout the Federal Government," FGMSD-76-5, Apr. 23, 1976.



14. "Improved Planning and Management of Information Systems Development Needed," LCD-74-118, Aug. 18, 1975.
15. "Opportunities for Improving Computer Use in the Bureau of the Mint," FGMSD-75-19, Mar. 20, 1975.
16. "Tools and Techniques for Improving the Efficiency of Federal Automatic Data Processing Operations," B-115369, June 3, 1974.
17. "Improving the Acquisition of Computer Systems," B-164031(4), Social Security Administration Department of Health, Education, and Welfare, Jan. 24, 1974.
18. "Acquisition and Use of Software Products for Automatic Data Processing Systems in the Federal Government," B-115369, June 30, 1971.

PRESIDENT'S REORGANIZATION PROJECT/  
OFFICE OF MANAGEMENT AND BUDGET

19. "Federal Data Processing Reorganization Study: Central Agencies Team Report" (Draft), July 5, 1978.
20. "Federal Data Processing Reorganization Study: Basic Report of the Science and Technology Team" (Draft), June 9, 1978.
21. "Federal Data Processing Reorganization Study: General Government Team Report" (Draft), May 16, 1978.
22. "Federal Data Processing Reorganization Study: Small Agencies Team Report" (Draft), May 1, 1978.
23. "Federal Data Processing Reorganization Study: Acquisition Team Option Paper," May 12, 1978.
24. "Federal ADP Reorganization Project Standards Study Team - Final Report," June 12, 1978.
25. "Federal Data Processing Reorganization Study: Operational Management Team Report," Sept. 1978.
26. "Information Technology And Governmental Reorganization - Summary of the Project" (Draft), Dec. 1978.
27. "Information Technology and Governmental Reorganization - Summary of The Federal Data Processing Reorganization Project," (Final Report), Apr. 1979.

NATIONAL BUREAU OF STANDARDS

28. "Appraisal of Federal Government COBOL Standards and Software Management: Survey Results," Donald R. Deutsch, NBSIR, 76-1100, Aug. 1976.
29. "Static Language Analysis," Gordon Lyon, NBS Technical Note 797, Oct. 1973.
30. "A FORTRAN Analyzer," Gordon Lyon and Rona B. Stillman, NBS Technical Note 849, Oct. 1974.
31. "Software Testing for Network Services," Rona B. Stillman and Belkis Leong-Hong, NBS Technical Note 874, July 1975.
32. "Six Data Base Management Systems: Feature Analysis and User Experiences," Elizabeth Fong, Joseph Collica, and Beatrice Marron, NBS Technical Note 887, Nov. 1975.
33. "Data Base Directions--The Next Steps," Edited by John L. Berg, NBS Special Publication 451, Sept. 1976.
34. "Computer Software Management: A Primer for Project Management and Quality Control," Dennis W. Fife, NBS Special Publication 500-11, July 1977.
35. "Software Tools: A Building Block Approach," I. Trotter Hardy, Belkis Leong-Hong, and Dennis W. Fife, NBS Special Publication 500-14, Aug. 1977.
36. "Computer Performance Evaluation Users Group," Edited by Dennis M. Conti and Josephine L. Walkowicz, NBS Special Publication 500-18, Sept. 1977.
37. "Audit and Evaluation of Computer Security," Edited by Zella G. Ruthberg, NBS Special Publication 500-19, Oct. 1977.
38. "Guide to Computer Program Directories," Compiled by Addie G. Chattic, NBS Special Publication 500-22, Dec. 1977.
39. "COBOL Instrumentation and Debugging: A Case Study," Gordon Lyon, NBS Special Publication 500-26, Jan. 1978.
40. "Guideline on Major Job Accounting Systems: The System Management Facilities (SMF) For IBM Systems Under OS/MVT," Garry Durbin, Todd Kinney, Peter Lamasney, Edward Newman, and Edward Syrett.

GENERAL SERVICES ADMINISTRATION

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



EXECUTIVE OFFICE OF THE PRESIDENT  
OFFICE OF MANAGEMENT AND BUDGET  
WASHINGTON, D.C. 20503

Mr. Donald L. Scantlebury  
Director, Division of Financial  
and General Management Studies  
U. S. General Accounting Office  
Washington, D.C. 20548

Dear Mr. Scantlebury:

This letter responds to your November 20th request for our views on your draft report entitled "Increased Use of Computer Software Tools and Techniques Offers Federal Managers Better Control and Cost Savings." We apologize for our delay in this response. We have recently reorganized the OMB functions dealing with this matter by establishing a new Office of Regulatory and Information Policy. An examination of some of our policies in this area has contributed to the delay.

With respect to your recommendation that OMB more clearly define the responsibilities of agency heads and ADP managers for the functions, we believe that OMB Circular No. A-71 generally covers these matters. However in light of your findings we plan to examine GSA and Commerce's activities in this area. Subsequent to this examination we shall determine whether our Circular should be modified.

We agree with GAO's conclusion that agencies are not always using the latest software technology. OMB has strongly encouraged the effective use of technology to improve Federal productivity and lower costs. The use of any technology, including software, however, must be justified. Agencies must evaluate the potential costs and identify quantifiable benefits. OMB will continue to support agency budget requests for the use of new technology whenever it is clearly justified.

Sincerely,

A handwritten signature in black ink that reads "Jim J. Tozzi".

Jim J. Tozzi  
Assistant Director for  
Regulatory and Information Policy



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:

Thank you for the opportunity to review and comment on the draft report entitled "Increased use of Computer Software Tools and Techniques offers Federal Managers better control and cost savings," dated November 26, 1979.

We basically agree with the report's findings and feel that there are significant savings to be achieved through increased use of these software tools/techniques by Federal managers.

The specific recommendations are summarized and discussed below:

RECOMMENDATION 1

That GSA "augment paragraph (3) of the Federal Property Management Regulation 101-35.206 to incorporate a list of actions that agencies should take to improve their application software and carry out the intent of the paragraph" (as suggested in Appendix I of report). We agree in principle with this recommendation, but reserve judgement as to the specific wording and location of the proposed change in the regulations.

RECOMMENDATION 2

"That GSA establish a set of standard tools for use in solving operational problems and promoting efficiency and economy." We support this recommendation. The National Bureau of Standards (NBS) has sent a questionnaire to approximately 500 Federal, academic and commercial ADP installations requesting information about the software tools used in each organization. Representatives of GSA met with NBS and reached agreement that NBS would forward the abstracts submitted by the agencies to the Software Exchange program for inclusion in the Federal Software Exchange catalog.

RECOMMENDATION 3

"Require that certain standard inspections, using software tools, be done on software developed for agencies by contractors." Again, we support the concept of a standard inspection or "system audit" of agency acquired

software. However, issues relating to contracting for computer software development were addressed in GAO Report B-115369 dated November 9, 1979, entitled "Contracting for Computer Software Development -- Serious Problems Require Management Attention to Avoid Wasting Additional Millions." These issues will be addressed in response to that report.

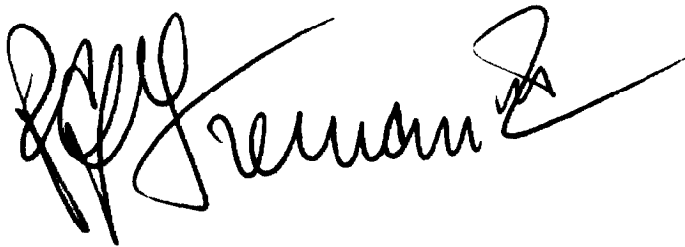
RECOMMENDATION 4

"To establish a software tools category at the Federal Software Exchange Center and provide the technical support necessary for sharing the tools." Establishing a set of standard tools, providing technical support and requiring the use of these tools on contractor developed software are potential functions for the new Software Development Office. As more personnel are hired and the functions of this new organization evolve, implementing this recommendation will receive strong consideration. We will include a software tools or software aids category in the next annual publication of the Federal Software Exchange Catalog.

The report further recommends to the Secretary of Commerce, that "NBS develop and adopt software tools and methods for inspecting contractor-developed software, and that GSA require their use in contracts." The latter part of this recommendation relates to GSA's contracting role. This portion of the recommendation will be addressed in the response to the aforementioned GAO report.

In conclusion, we support the GAO draft report and will implement the report's findings to the maximum extent possible.

Sincerely,



R. G. Brennan III  
Administrator



**UNITED STATES DEPARTMENT OF COMMERCE**  
**The Assistant Secretary for Science and Technology**  
Washington, D.C. 20230  
(202) 377-3111

Mr. D. L. Scantlebury  
Director, Financial and General  
Management Studies Division  
General Accounting Office  
441 G Street, NW  
Washington, DC 20548

Dear Mr. Scantlebury:

Thank you for the opportunity to provide comments on the draft report to Congress on "Increased Use of Computer Software Tools and Techniques Offers Federal Managers Better Control and Cost Savings." I concur with the basic intent of the report. However, the recommendation for NBS action seems ambiguous and inconsistent.

On page iii, NBS is asked to "develop or adopt standards of guidelines for using software tools," and on page 39 "to develop or adopt software tools and methods..." The difference between developing tools and developing usage guidelines is considerable, both in meaning and in resources required for compliance. Clarification in this matter would be most welcome.

In the general area of software tools, it may be helpful to note that NBS is already in the process of developing a program analyzer for programs written in the new proposed Federal standard FORTRAN programming language. This software tool will be the first standard tool developed under the NBS Software Quality task. Software tools are complex and expensive to produce and maintain, however. The present NBS program does not have the funding or staff to do more than demonstrate the feasibility of such tools through prototype efforts.

Several other activities that relate directly to the concerns of this report have been initiated at NBS as part of its strengthened ADP Standards Development Program. These are described below in the context of those sections of the report to which they apply.

1. GAO discusses on page 27 the current need for standards for evaluation of the effectiveness of software tools and for standards for the use of software tools. Two specific efforts are currently under way at NBS to attack this need. The first is to develop a classification system for software tools. Such a classification system will allow NBS to categorize currently available tools. The

second effort involves the definition of various programming environments (i.e., scientific, business, command and control, etc.) that determine the types of software tools needed.

2. GAO recommends on page 38 that OMB direct the establishment of a coordinated Government-wide research effort for software tools that will include dissemination of information to all potential Federal users. NBS has initiated a task in this area, and a press release has been sent to major publications requesting participation in an information exchange on software tools. To date NBS has collected information on over 600 software tools. NBS is placing this data into a data base system from which information may be readily extracted and made available to Federal agencies in printed form.
3. GAO recommends on page 38 that GSA require that certain standard inspections be done of software developed for the Federal Government. NBS is developing guidelines for validation and verification of computer software for the various Federal ADP environments.

The following specific comments are offered to correct or clarify other sections of the report:


1. The description of the software life cycle on page 2 for applications programs does not agree with the life cycle specified in FIPS PUB 38 and FIPS PUB 64.
2. Terminology is sometimes inconsistent or unorthodox. For example, the use of the term "filter" on page 5 is confusing. The type of preprocessor discussed is commonly called an auditor or checker. The definitions of "Program Analyzer" on pages 5 and 23 do not quite agree.
3. Structured programming (page 6) is much more than a technique of arranging statements. Structured programming is a technique used to develop computer programs so they can be more easily documented, updated, corrected, tested, and understood.
4. Structured walk-throughs should be included as an example of software techniques (page 6).
5. There is much emphasis on optimization in this report. The report should also point out, however, that program optimization carried out without adequate regard for software portability and maintenance requirements may obscure the structure of a computer program and make it more expensive to maintain and more difficult to convert to other machines. The savings gained by unchecked program optimization is often lost in increased maintenance costs and less tool portability.



6. The second paragraph of page 21 exaggerates the capability of software tools and overstates the current state of the art. It would be better to say that software tools can provide assistance in the preparation of test data and in determining what logic of a program has not been exercised.
7. The second paragraph of page 35 discusses duplication of Government efforts to develop a software cost estimation model. These efforts are not duplications because they are aimed at different programming environments.

I hope you will find these comments useful in preparing the final report which I look forward to seeing when it is ready.

Sincerely,



Jordan J. Baruch

DETAILED RESPONSE TO COMMERCE COMMENTS

NBS offered several detailed comments "to correct or clarify other sections of the report." They start on the second page of the NBS comments reproduced in appendix III.

- "1. The description of the life cycle for application software in Chapter 1 does not agree with the new life cycle specified in FIPS PUB 1/ 38 and FIPS PUB 64."

We know that the FIPS PUBS 38 and 64 speak of the life cycle in terms of an initiation phase, a development phase, and an operation phase. However, the American National Standard Dictionary For Information Processing (X3/TR-1-77, adopted as a Federal Standard with FIPS PUB 11-1 dated Sept. 30, 1977) does not have any definitions of "software development," "software life cycle," or, indeed, "life cycle" itself. Thus, there is apparently no universal Federal definition of the software life cycle, so we felt free to coin our own. In explaining applications software to our lay audience, we wanted to point out that there are activities done before software produces user output 2/--our word "development," and "initiation and development" in the FIPS publications--and activities done after the software produces its first user output--our "operational or production" phase, called the "operation phase" in the FIPS publications.

- "2. Terminology is sometimes inconsistent or unorthodox. For example the use of the term 'filter' on page 5 is confusing. The type of preprocessor discussed is commonly called an auditor or checker. The definitions of 'Program Analyzer' on pages 5 and 23 do not quite agree."

First, we do not understand NBS' use of the term "unorthodox" because there is apparently no standard. The American National Dictionary for Information Processing (X3/TR-1-77 FIPS 11-1 referred to above) does not contain the terms "auditor," "code auditor," "checker," or "standards checker," 3/ and does define "filter" as a "device or program that separates data signals or material in accordance

---

1/Federal Information Processing Standards Publication.

2/The checks printed by a payroll program are an example of user output.

3/All have been used in the literature.

with specified criteria." <sup>1/</sup> Since a preprocessor processes other programs as "data \* \* \* or material" we feel that our use of the word does not violate the "orthodox" definition. Also, in our discussion, we enclosed the word "filter" in quotation marks to indicate the function to a lay reader. To reduce possible confusion we have added the phrase "(also known as code auditors)" to our discussion of software tools in chapter 1.

Our first definition of program analyzer was more inclusive than the second one which was inserted to remind the reader. For perfect agreement, we have made both the same.

- "3. Structured programming (page 6) is much more than a technique of arranging statements. Structured programming is a technique used to develop computer programs so that they can be more easily documented, updated, corrected, tested and understood."

Here again, we used a brief definition for the lay reader, and, again the American National Dictionary (FIPS 11-1) does not contain a definition. The literature abounds with various definitions of structured programming, all of which indicate that it yields programs which are easier for humans to work with. We believe that our draft definition implied the activities listed by NBS.

The draft said:

"Structured programming, also called structured coding: a technique of arranging the actual statements of computer programs so that they will be more easily understood by others who must later maintain and modify them."

However, we changed the wording to that now shown in the software techniques section of chapter 1.

- "4. Structured walkthroughs should be included as an example of software techniques (page 6)."

Our list is not intended to be all-inclusive--our introductory statement reads "Examples of software techniques useful to workers include:" We know of many other techniques including egoless programming, bottom-up testing, top-down testing, unit testing, stubs, and test drivers. Since the

---

<sup>1/</sup>GAO emphasis added.

writer judges what shall be included in a brief noninclusive list, we did not add structured walkthroughs.

- "5. \* \* \* The report should also point out that program optimization \* \* \* may obscure the structure of a computer program and make it more expensive to maintain and more difficult to convert to other machines. The savings gained by unchecked program optimization is often lost in increased maintenance costs and less tool portability."

We are aware that many changes made to computer programs in the name of optimization may conflict with other goals such as the programs' ability to be maintained and to be converted to replacement computers. For example, an excuse sometimes given for not using structured programming is that it is "less efficient in the machine." On this latter point, recently published findings 1/ indicated that structured programming does not necessarily conflict with machine efficiency. Generally, many changes can be made to computer programs to reduce their machine operating costs without adding to maintenance costs. We agree with NBS, however, that changes made to computer programs to reduce their machine operating costs must be evaluated carefully to ensure that they do not increase other costs, including maintenance, so much that the machine savings are lost. To clarify this matter, we added some discussion of it to the discussion of reducing operating costs in chapter 3.

- "6. The second paragraph of page 21 exaggerates the capability of software tools and overstates the current state of the art. It would be better to say that software tools can provide assistance in the preparation of test data and in determining what logic of a program has not been exercised."

Our draft said "Software tools can automate much of the work of preparing test data and of verifying that the test data has indeed caused all the logic of the programs to be exercised." We perceived that our draft wording is both a true statement and synonymous with that proposed by NBS. However, we have changed the wording to say "reduce the labor

---

1/Including a paper, "Optimizing Program Quality and Programmer Productivity" by Capers Jones of IBM. Tom Peters of IBM presented the paper at the 50th session of SHARE, the IBM users group, March 7, 1978.

involved in"--certainly true and, we believe, stating even more clearly that tools alone do not do the whole job.

- "7. The second paragraph of page 35 discusses duplication of Government efforts to develop a software cost estimation model. These efforts are not duplications because they are aimed at different programming environments."

We realize that the studies were aimed at different programming environments and feel that there was, nevertheless, some duplication.

(913380)



Single copies of GAO reports are available free of charge. Requests (except by Members of Congress) for additional quantities should be accompanied by payment of \$1.00 per copy.

Requests for single copies (without charge) should be sent to:

U.S. General Accounting Office  
Distribution Section, Room 1518  
441 G Street, NW.  
Washington, DC 20548

Requests for multiple copies should be sent with checks or money orders to:

U.S. General Accounting Office  
Distribution Section  
P.O. Box 1020  
Washington, DC 20013

Checks or money orders should be made payable to the U.S. General Accounting Office. NOTE: Stamps or Superintendent of Documents coupons will not be accepted.

**PLEASE DO NOT SEND CASH**

To expedite filling your order, use the report number and date in the lower right corner of the front cover.

GAO reports are now available on microfiche. If such copies will meet your needs, be sure to specify that you want microfiche copies.

**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**