BY THE U.S. GENERAL ACCOUNTING OFFICE

Report To The Administrator Of General Services

Greater Emphasis On Testing Needed To Make Computer Software More Reliable And Less Costly

Federal agencies spend billions of dollars annually on computer software programs. However, unless these programs receive adequate testing, federal managers cannot be reasonably sure that they provide accurate, reliable information and that computer resources are used efficiently and economically.

GAO found that agencies do not manage the software testing process and, as a result, problem software, once in operation, is unreliable and costly.

The General Services Administration has already taken some steps to improve the quality of software, but more remains to be done. Heads of federal agencies should require automatic data processing managers to place greater emphasis on software testing.

122852

122852
027198

GAO/IMTEC-84-2
OCTOBER 27, 1983

Request for copies of GAO reports should be sent to:

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

Telephone (202) 275-6241

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

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

INFORMATION MANAGEMENT
& TECHNOLOGY DIVISION

B-206180

The Honorable Gerald P. Carmen
The Administrator of General Services

Dear Mr. Carmen:

This report discusses the adequacy and effectiveness of federal agencies' software testing practices.

We found that many improvements are needed in (1) the planning for ADP software testing, including defining the testing procedures, criteria, and techniques required before placing either agency- or contractor-developed software into operation; (2) the process by which the agencies monitor and enforce compliance with testing objectives and strategies; and (3) the utilization of automated tools and testing techniques.

We are sending copies of this report to the heads of federal agencies.

Sincerely yours,

Warren G. Reed
Director

GENERAL ACCOUNTING OFFICE
REPORT TO THE ADMINISTRATOR
OF GENERAL SERVICES

GREATER EMPHASIS ON TESTING NEEDED TO MAKE COMPUTER SOFTWARE MORE RELIABLE AND LESS COSTLY

D I G E S T

Testing is a critical process in developing and maintaining computer programs. The purpose of testing is to detect errors before the programs are put into operation. Inadequate testing increases the potential for undetected errors and reduces the extent to which the software can be relied on to safeguard assets and provide accurate information.

Considering the billions of dollars the federal government spends on software and agencies' reliance on it to perform their missions, software testing does not receive appropriate management emphasis. GAO based its findings on visits to eight federal agencies and automatic data processing (ADP) installations and detailed reviews of business application software--a program which performs accounting, payroll, and related functions. It also sent questionnaires to 600 randomly selected federal ADP installations to obtain information on the status of software testing at those installations. That information was then used to project the status of software testing government-wide.

The Brooks Act (Public Law 89-306) assigned certain responsibilities for automatic data processing to the General Services Administration, the National Bureau of Standards, and the Office of Management and Budget. The General Services Administration is responsible for developing, implementing, and monitoring government-wide policy for the acquisition, use, and management of ADP resources. The National Bureau of Standards is responsible for providing scientific and technological advisory services and for developing Federal Information Processsing Standards. The Office of Management and Budget is responsible for fiscal and policy control. In addition, each federal agency has certain responsibilities for managing its own ADP resources.

AGENCIES NEED TO BETTER MANAGE THE SOFTWARE TESTING PROCESS

GAO's review showed that agencies do not always adequately manage the overall testing process to help ensure its effectiveness in producing accurate and reliable software. GAO observed testing deficiencies at the eight agencies and installations visited. In addition, responses to questionnaires from 207 of the installations surveyed indicate that these deficiencies are government-wide.

Poor management practices GAO observed included the following:

--Agencies do not always enforce testing policies and requirements. For example, required unit and system testing for a payroll modification was omitted because the programmer considered the change minor. Corrective action for the resulting error in this program required $10,000 for automatic data processing costs and caused agency field offices to manually review about 5,000 pay accounts for potentially incorrect payments. In addition, GAO's survey of data processing installations showed that other testing-related requirements were not enforced. For instance, 73 percent of the installations reported they did not document their computer programs according to National Bureau of Standards guidelines. These guidelines include the requirement that a test plan and a test analysis report be prepared. (See pp. 6-7.)

--Agencies do not always provide those responsible for testing with written guidance containing the specific procedures, criteria, and techniques required for testing agency software. GAO's survey of data processing installations showed that only 44 percent said they had received written guidance on testing of business application software from their department or agency. (See pp. 7-8.)

--Agencies do not always collect and/or evaluate data on software testing problems to improve the testing process. GAO's survey of data processing installations confirmed that at least 65 percent do not maintain such data. (See pp. 9-10.)

--Agencies do not allow adequate time for the planning and testing of business application software. GAO found examples where the planned time allotted for testing was insufficient, so in order to deliver the products on time, testing was reduced. GAO's survey of data processing installations also showed that at least 29 percent believed users did not allocate enough time for testing in the development process. (See pp. 10-12.)

--Agencies do not always require the use of software tools and techniques--computer programs to test the thoroughness and efficiency of software testing in the testing process. GAO found that only one of the eight agencies reviewed required that software tools be used in the testing process. GAO also found that only 13 percent of the data processing installations surveyed used software test tools. (See pp. 12-14.)

Management must be able to rely on the testing process to help ensure that the internal controls in computer software are adequate and operating properly. (See p. 14.)

POORLY TESTED SOFTWARE IS COSTLY AND UNRELIABLE

According to an industry study, costs to correct errors after software is put into operation may be over 7 times higher than if detected during unit testing, and 30 times higher than if detected during design. (See pp. 21-22.) For example, just one payroll system error required an agency to review thousands of pay accounts and make corrective payments manually. These errors also

lead to breakdowns in control over financial assets. For instance, one agency's automated contract administration system exceeded contract progress payment limits by more than $500,000 because of a programming error. The Government lost interest on these funds and in one case had to recover an overpayment of about $44,000 from a contractor. (See p. 16.) These examples demonstrate that poor testing increases the potential for error and thus reduces software reliability. Although testing cannot be expected to detect all errors, the number of test cases and conditions should be adequate to reasonably ensure that software is error free. (See pp. 15-20.)

GAO also found that poor testing against user needs, as well as limited user involvement in the testing process, contributed to costly software problems. Such testing and user involvement helps ensure that application systems will satisfy current user needs and perform as intended. As a result of poor testing, one agency failed to detect that a system would not perform as the user intended. Consequently, once it was in operation the system had to be redesigned at additional cost. (See p. 20.)

RECOMMENDATIONS

GAO recommends that the heads of federal agencies:

--Establish written software testing policies and requirements defining the testing procedures, criteria, and techniques required before placing either agency- or contractor-developed software into operation. These should include specific requirements for user participation in the testing process.

--Monitor and enforce compliance with these testing policies and requirements.

--Periodically evaluate the software testing process to determine (1) its effectiveness in preventing errors and reducing costs associated with error correction and (2) the appropriate allocation of staff and computer resources to software testing.

--Identify and incorporate into the testing process those automated tools and testing techniques that can help the agency provide more thorough testing and more efficient resource use. This should include providing appropriate training on these tools and techniques.

GAO also recommends that the Administrator of General Services, through the Office of Software Development, review selected software development projects in Federal agencies to identify uses and potential uses of software tools and techniques that improve testing thoroughness and efficiency. The Office should then report on these reviews to provide guidance for agencies in incorporating tools and techniques into their testing processes.

AGENCY COMMENTS AND OUR EVALUATION

GSA is in complete agreement with this report and has, under its new Office of Software Development, taken steps to improve the quality of software and its testing by assisting agencies in accepting and using state-of-the-art software technology. In view of the billons of dollars spent annually on software, however, GAO believes even more remains to be done in improving this area. (See app. V.)

Contents

		Page
DIGEST		i
CHAPTER		
1	INTRODUCTION	1
	Software testing defined	1
	Earlier GAO reports note problems with poorly tested software	2
	Roles of various agencies	3
	Objectives, scope, and methodology	3
2	AGENCIES NEED BETTER MANAGEMENT OF THE SOFTWARE TESTING PROCESS	5
	Agencies do not enforce testing requirements	5
	Agencies need supplemental testing guidance	7
	Agencies do not use software problem data to evaluate testing	9
	Managers can minimize the effects of time and resource constraints on testing	10
	Software testing helps ensure adequate internal control systems	14
3	POOR TESTING RESULTS IN COSTLY AND UNRELIABLE SOFTWARE	15
	Costly errors and reduced software reliability result from poor testing	15
	Better testing against user requirements is needed to ensure that software meets user needs	20
	Testing reduces the cost of error correction	21
4	CONCLUSIONS AND RECOMMENDATIONS	23
	Conclusions	23
	Recommendations	24
	Agency comments and our evaluation	24

APPENDIX

I	Agencies and installations visited and surveyed	26
II	Case studies	29
III	Summary of questionnaire results	42
IV	Sampling methodology, data collection, quality control, and projected results	57
V	September 14, 1983, letter from the Administrator of General Services	64

ABBREVIATIONS

ADP	automatic data processing
FIPS	Federal Information Processing Standards
GAO	General Accounting Office
GSA	General Services Administration
NBS	National Bureau of Standards
OMB	Office of Management and Budget

CHAPTER 1

INTRODUCTION

Software--the programs or sets of instructions that run a computer--is the most expensive component of computer resources. Industry sources predict that by 1985 about 90 percent of automatic data processing (ADP) costs will be attributable to software. For the federal government, whose ADP costs are currently more than $15 billion annually, this represents a sizable investment. Application programs automate the tasks of end users, including such business-related tasks as payroll or such scientific tasks as simulations. Adequate testing of applications software can help federal managers be reasonably assured that software will correctly automate user needs and produce accurate and reliable results. It can also help minimize the resources needed to correct both software errors and their effects.

This report discusses our review of software testing at several federal agencies and installations, illustrates the costs and effects of inadequate testing, and recommends improvements.

SOFTWARE TESTING DEFINED

Software testing is the process of identifying program errors by analysis or by executing a program on a computer using actual or test data. Developing application software is a labor intensive, error-prone process, and errors can be made both in deciding what the programs should do and in writing them to do it. The objective of testing is to find and correct these errors before the software is put into operation to do the user's work.

Testing can be either manual or automated. It can focus on any of various software aspects including the requirements for the software, the design specifications, the individual programs, or the overall system, which typically is several programs. Manual testing techniques usually involve comparing the product of the software against a list of test criteria, which include checks for common errors. One manual test method is desk checking, where the programmer or another person individually checks the program code. An inspection is similar to desk checking except that a team, rather than an individual, checks the software. In another method--the walkthrough--the programmer explains the system logic to a team using test data.

In most automated testing, software is actually run on the computer and the output is compared with expected results. Generally, the test data include both invalid and unexpected data as well as valid and expected data. Some types of automated testing include:

--Unit test. Testing an individual program.

--Integration test. Testing the several programs of a system to see if they work together correctly.

--System test. Usually, testing to determine whether the system meets user requirements and objectives.

--Acceptance test. Testing, usually performed by the user, to compare the program or system with its initial requirements and current user needs.

EARLIER GAO REPORTS NOTE PROBLEMS WITH POORLY TESTED SOFTWARE

Past government-wide GAO reports have expressed our concern with the lack of adequate software testing. In a 1979 report about contractor-developed software, we pointed to the need for better definition and enforcement of acceptance testing requirements for contract work.[1] A 1980 report recommended that the General Services Administration (GSA) require contractor-developed software to pass a standard inspection process which uses automated software tools for testing and analysis.[2] The report also recommended that OMB require all federal agency heads to consider software quality assurance in their agencies and, where cost beneficial, establish an ongoing quality assurance function independent of software developers, which the report said could be made part of an agency's internal audit organization. And in 1981, a report on federal agencies' maintenance of contract programs recommended that new software be tested more thoroughly to remove defects in program logic and thus help reduce the cost of maintenance work to fix defects.[3]

Other reports issued on individual agencies' computer systems noted programming problems that caused excessive data errors, increased operating costs, and allowed millions in erroneous entitlement payments to go unrecovered.[4]

---

[1] "Contracting for Computer Software Development--Serious Problems Require Management Attention to Avoid Wasting Additional Millions" (FGMSD-80-4, Nov. 9, 1979).

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

[3] "Federal Agencies' Maintenance of Computer Programs: Expensive and Undermanaged" (AFMD-81-25, Feb. 26, 1981).

[4] "The Marine Corps Military Pay System: Too Many Errors and Inefficiencies" (FGMSD-80-49, June 10, 1980) and "The Social Security Administration Needs To Develop a Structured and Planned Approach For Managing and Controlling the Design, Development, and Modification of Its Supplemental Security Income Computerized System" (HRD-80-5, Oct. 16, 1979).

Although we do not know the full extent to which federal agencies took action on our recommendations in the above noted reports, we do know that GSA, the National Bureau of Standards (NBS), and the Office of Management and Budget (OMB) have taken appropriate action on some or all of the recommendations.

We are issuing this report to show the effects of poor testing on the federal government's business application software. In general, it recommends that federal agencies establish and enforce software testing policies and requirements, and periodically evaluate their effectiveness.

ROLES OF VARIOUS AGENCIES

The Brooks Act (Public Law 89-306), enacted in October 1965, provides for the economic and efficient purchase, lease, maintenance, operation, and use of ADP equipment. The responsibilities under the act are assigned to several agencies. GSA is responsible for developing, implementing, and monitoring government-wide policy for the acquisition, use, and management of ADP resources. The Department of Commerce, primarily through NBS, is responsible for providing scientific and technological advisory services and for developing Federal Information Processing Standards. OMB is responsible for fiscal and policy control. In addition, each federal agency has certain responsibilities for managing its own ADP resources.

In our role of aiding the Congress, we are concerned with the management of federal ADP resources. Our past reports to the Congress have recommended improvements in ADP management both government-wide and at specific agencies.

OBJECTIVES, SCOPE, AND METHODOLOGY

The objective of our review was to determine the adequacy and effectiveness of software testing practices federal agencies use to obtain business application software. We focused on business application software because it represents more than 60 percent of the federal government's computer programs, and because it automates the processing transactions that affect the government's financial resources and management information. From October 1981 to August 1982 we visited a total of eight selected federal agencies and installations that depend on ADP to perform their mission. (See app. I.) We reviewed their software testing policies and procedures, and, where apropriate

--reviewed and analyzed logs containing information on ADP system failures and programming errors and

--interviewed both ADP personnel and application software users to discuss agency testing policies and procedures and testing-related problems.

We also selected for analysis a total of 12 major business application systems at the agencies and installations visited. For each of these systems, we analyzed the specific software testing the agencies conducted and any programming errors that occurred after a system became operational. From these analyses we noted instances in which poor testing practices had failed to detect software errors. However, these examples do not necessarily imply inadequate testing for all systems maintained at the specific data processing installation or agency.

We used a questionnaire to survey 600 randomly selected federal ADP installations. Of the 477 responses received, 207 reported at least one business application program. These 207 questionnaire responses represent 37 agencies. Based on this response, we estimated that 1,526 of the 4,423 installations in our universe have at least one business application program, and projected our survey results to this number. Appendix III (see p. 42) presents an overall summary of the survey results, and appendix IV (see p. 57) provides our sampling methodology.

Of the three central agencies under the Brooks Act, we solicited formal comments from GSA because the majority of our findings relate to GSA's specific role in providing assistance to agencies in accepting and using state-of-the-art software technology. In addition, we considered in our report the informal comments of these agencies and installations on the case studies we developed.

We agreed that the agencies and installations would not be identified in this report, as the case studies represent typical situations that are frequently encountered and are presented only to illustrate their nature.

We performed this review in accordance with generally accepted government audit standards.

CHAPTER 2

AGENCIES NEED BETTER MANAGEMENT OF THE SOFTWARE TESTING PROCESS

Agencies do not manage software testing effectively, considering the importance of testing in ensuring software accuracy and reliability. This is at least partly because agencies do not always (1) enforce their testing requirements, (2) give their staff written guidance on testing policies and requirements, (3) use data on software problems as feedback on the effectiveness of the testing process, or (4) allow adequate time for planning and testing. Consequently, not all software is adequately tested, and managers cannot rely on the testing process to help assure that internal control systems are appropriate and working properly.

AGENCIES DO NOT ENFORCE TESTING REQUIREMENTS

Most agencies have some specific procedures and techniques for testing software. However, we found that systems development managers generally did not measure or enforce compliance with these requirements. Therefore, these managers cannot be sure that agency software consistently receives the required degree of testing.

The software testing procedures and techniques agencies require include:

--Preparing written test plans and documenting test results.

--Performing specific types of testing (such as unit, systems, or acceptance testing).

--Using specific testing techniques (such as walkthroughs, desk checks, and reviews).

--Using automated test tools (that is, computer programs that help test other programs).

Nevertheless, systems development staff sometimes decide for themselves what testing procedures or techniques will be used, with little review by systems development managers. We found instances where required test plans were not prepared, walkthroughs and reviews were not performed, unit and system tests were omitted or curtailed, and required tests using automated testing tools were not administered.

In some cases these omissions contributed to testing failure, and errors in the software were not detected before it became operational. For instance, a programmer considered a payroll system modification minor and chose to omit required unit and system testing of the change. Managers did not enforce the testing requirement, and after only limited group testing the modi-

fied program was put into operation. The program proved to have an error and it cost $10,000 to correct and caused agency field offices to manually review about 5,000 pay accounts for potentially incorrect payments.

In another example, an agency's certification requirement was not enforced. The agency requires that all new and modified programs be certified by a quality assurance staff before system testing. The agency initiated the requirement in 1977; however, at the time of our review, not all programs had been submitted for certification. A member of the quality control staff and a system development official admitted that no one knows and they could not estimate how many of the agency's programs should have been submitted for certification. The quality control staff member also explained that managers in different divisions of the system development group did not uniformly emphasize compliance with certification requirements, and the quality assurance group did not enforce the requirements. As a result, many programs have not been certified.

Our survey of ADP installations also revealed that agencies apparently do not enforce testing requirements. National Bureau of Standards (NBS) guidelines for documentation of computer software recommend that a test plan and a test analysis report be prepared.[1] These documents help managers determine whether testing requirements have been met.[2] However, as the table of survey responses below indicates, at least 73 percent of the installations reported they do not have the required documentation.

---

[1] "Guidelines for Documentation of Computer Programs and Automated Data Systems," Federal Information Processing Standards Publication (FIPS PUB) 38; Feb. 15, 1976.

[2] According to the guidelines, the test plan should specify such things as the testing milestones, schedule, and resource requirements; the specifications, descriptions, and procedures for all tests; and rules for evaluating test results. The test analysis report should identify the results and findings of each test including any deficiencies noted and recommendations for corrections.

Programs documented	Installations (note a) Number	Percent
All or almost all programs	111	7
Most programs	52	3
About half the programs	22	1
Some programs	118	8
Few or no programs	1,105	73
No answer	118	8
Total	1,526	100

a/Based on a projected total of 1,526 installations with at least one business application program.

The examples discussed above show that although testing requirements exist, they are not always enforced. Without this enforcement, neither system development managers nor agency managers can be sure that agency software receives adequate testing.

AGENCIES NEED SUPPLEMENTAL TESTING GUIDANCE

Agency testing guidance is needed to establish consistent software policies and requirements agencywide and to communicate these to ADP staff, users, and others in the agency. Our survey showed at least 39 percent of the installations had received NBS guidance on software testing, while at least 48 percent had not; 13 percent did not respond to the question. Specific NBS guidance on software testing currently includes:

--"Computer Software Management: A Primer for Project Management and Quality Control," NBS Special Publication 500-11; July 1977.

--"Validation, Verification, and Testing of Computer Software," NBS Special Publication 500-75; February 1981.

--"Planning for Software Validation, Verification, and Testing," NBS Special Publication 500-98; November 1982.

--"Software Validation, Verification, and Testing Technique and Tool Reference Guide," NBS Special Publication 500-93; September 1982.

In addition, as discussed on page 6, NBS issued FIPS PUB 38, which includes guidelines for criteria and content in preparing software test plans and test analysis reports. However, we found that some of the agencies we reviewed developed their own guidance in addition to the general guidance NBS provided. Our survey indicated that at least 45 percent of the installations have not received testing guidance from their agencies, while 44 percent have; 10

percent did not respond to the question. Those who have received agency guidance consider it more useful than other central testing guidance.

Several agencies we reviewed had developed formal guidance including agency policies, requirements, and standards for software testing. For example, one systems development group prepared guidelines on the procedures necessary to test a new or modified system, from the initial tests of individual programs through completion of system testing. The guidelines, to be used in conjunction with other agency ADP manuals, govern test preparation, execution, and review. They discuss the roles in the testing process of the programmer, the systems analyst or computer specialist, the development project leader, and the user. Also included are requirements to use specialized testing programs, considerations to note in preparing for and evaluating system testing, and guidance for generating test data.

Our survey indicated that testing guidance provided at the agency level may be more useful than that provided by NBS. Forty-four percent of the installations reported they had received some written guidance from their department or agency on testing business application software. Of these, 81 percent indicated that this guidance was of from moderate to very great use; to 48 percent of those, it was of great or very great use. This contrasts with the same installations' opinions of NBS guidance. Only 43 percent of the installations receiving NBS guidance indicated it was of from moderate to great use; and to only 10 percent of those was it of great to very great use. The table below compares these survey results.

Installations' evaluation of use	Agency guidance		NBS guidance	
	Number	Percent	Number	Percent
Very great use	162	24	22	4
Great use	162	24	37	6
Moderate use	221	33	192	33
Some use	74	11	206	35
Little or no use	15	2	111	19
No answer	44	7	22	4
Total	678	a/ 100	590	a/ 100

a/Total does not add due to rounding of data.

Agency testing guidance can set the specific testing policies and requirements for the agency. This guidance can help ensure that the testing procedures, criteria, and techniques considered necessary for adequate testing are used consistently for all agency software. For example, one agency we reviewed had not developed agencywide guidance, and we found that separate systems development groups within the agency used different procedures and techniques to test their software. One group required walkthroughs and one

did not; one used a standing user acceptance committee and the other did not. While this does not necessarily indicate inadequate testing by either group, these different procedures and approaches make it difficult to assess the appropriateness of the testing that was done.

AGENCIES DO NOT USE SOFTWARE PROBLEM DATA TO EVALUATE TESTING

Agencies do not use data on software problems in operational systems to help evaluate the testing process. Our review indicated that the eight agencies we reviewed did not keep adequate data on such problems or use available data to provide feedback on software testing. Such feedback can help systems development managers determine the adequacy and effectiveness of existing testing policies and requirements.

Information on the type and frequency of software errors can indicate strengths and weaknesses either in the testing of a particular project or in the overall testing process. Other information, such as the cost to repair software errors, can also show the cost impact of inadequate testing. Despite the usefulness of this information, our survey showed that most installations do not maintain this type of data. (See p. 50.) The following table, which summarizes our survey results, shows data maintenance practices of the projected installations for fiscal year 1981.

1,526 Projected Installations

	Maintained		Not maintained		No answer	
Type of data	Number	Percent	Number	Percent	Number	Percent
Types of software failures	472	31	1,010	66	44	3
Frequency of software failures	450	30	1,039	68	37	2
Cost to repair failures	170	11	1,297	85	59	4

The agencies and installations we visited also did not generally maintain data on software problems and errors for analysis. However, some agencies did have systems or procedures to account for software development activities that could be used to provide feedback on the testing process. For example, a systems development group at one agency had an automated project control system which contained information on its work projects, including those initiated to correct programming errors. Using this system, we identified and analyzed programming errors related to a specific, ongoing modification project. Our analysis showed that in the year following initiation of this project, about 980 staff hours were needed to correct related programming errors. This amounted to about one-fourth of the time used originally to make

modifications--about 3,800 hours. Systems development managers had not used the project control system to provide this type of data, but agreed that such feedback could be useful in evaluating testing processes.

In another agency, ADP problems and software errors are recorded in (1) formal requests for assistance from the user and (2) other reports of ADP difficulties prepared within the ADP group. Information from user requests is summarized monthly by type of problem and by computer system, but systems development managers did not use this information to provide feedback on software testing.

To demonstrate the potential use of this information, we analyzed data on user assistance requests initiated from July 1981 to March 1982. For one of the systems reviewed at this agency, our analysis showed that over a 9-month period, 30 assistance requests were classified as system design errors. In discussing our analysis, a systems development project leader and a user representative acknowledged that problems described in some of these assistance requests were undetected in testing because test data and results developed for particular modifications to this system were not comprehensive.

These examples show that collecting and analyzing data on software errors can help managers evaluate the effectiveness of testing in preventing or reducing such errors. We believe that doing so is essential in devising adequate software testing policies and requirements.

MANAGERS CAN MINIMIZE THE EFFECTS OF TIME AND RESOURCE CONSTRAINTS ON TESTING

The amount of software testing done may be restricted by time and schedule constraints or by the availability of trained staff. However, such restrictions do not eliminate the need for adequate testing. Managers should make the most of available resources through adequate test planning and proper allocation of staff. Other measures include increasing productivity by providing software testing tools and by training staff in the use of testing tools and techniques.

Project deadlines and development delays cause reduced testing

Much of testing takes place in the latter stages of the software development or modification process. Therefore, when development or modification work falls behind schedule and the user's requirements make the planned delivery date relatively inflexible, testing is sometimes reduced to permit timely delivery. For instance, one agency implemented modifications to an employee life insurance system despite known errors and untested conditions. The agency chose to do so because changes in the law

covering the insurance program had to be incorporated as soon as possible to minimize the effect of the changes on the agency's annuity system.

In another example, a systems development group implementing a major payroll modification asked the payroll department for test data. The data were not provided in time to meet payroll processing deadlines, and the development group was required to deliver the modification without first testing this data. While we are not aware of any errors as a result of the reduced testing, these examples show that testing can be slighted due to emphasis on delivery schedules.

Failure to provide sufficient time for testing in the development schedule is another reason why the software testing that is done might be insufficient. Our survey of data processing installations indicated that at least 29 percent of the installations believed users did not allocate enough time in the development process for testing. (See p. 49.)

Time and schedule restrictions do not eliminate the need for adequate testing. The Variable Housing Allowance system discussed on page 16, is an example of how errors can go undetected even when a system was tested, if that testing was inadequate. Agency management directed that the system be made operational without complete testing because further delay would result in nonpayment of the allowance or would require use of expensive manual payment procedures. This incomplete testing failed to detect a software error that resulted in 750 incorrect pay accounts and required manual corrections. The agency believes that, in this instance, implementing the system was more practical than completing testing. However, we believe that this example shows the potential consequences of curtailed testing, and that proper test planning and scheduling can decrease demands to compromise testing.

Staff shortages reduce testing

Another factor that affects software testing is the availability of staff. At some agencies we visited and in response to our questionnaire, agency systems development managers cited staff shortages as a testing constraint. For example, a systems development manager at one agency believed that about 50 percent of software development staff time should be devoted to testing, but due to staff shortages he budgets only 25 percent. Also, 41 percent of those responding to our survey felt that staff resources available for software testing were less than adequate. The table below summarizes those responses.

Response	Installations Number	Installations Percent
More than adequate	272	18
Adequate	590	39
Less than adequate	619	41
Don't know	44	3
Total (note a)	1,526	100

a/ Totals do not add due to rounding and weighting of data.

Use of automated tools and other techniques can increase testing productivity and effectiveness

Two agencies we reviewed used software tools[3] in the testing process, but our survey indicated that most installations do not. Moreover, most of these installations indicated a need for additional training both in the use of software tools and in testing techniques. Use of tools and techniques can increase efficiency and effectiveness, and therefore compensate to some extent for time and resource constraints.

Tools useful for software testing include test data generators and test coverage analyzers. A test data generator analyzes a program or expected program input and generates test data. A test coverage analyzer monitors a program while a set of a test data is being executed, and measures the percentage of program logic the test executes. A more detailed list and description of software testing tools can be found in NBS Special Publication 500-93, "Software Validation, Verification, and Testing Technique and Tool Reference Guide," September 1982. Another source of information on software tools is the Federal Software Testing Center of GSA's Office of Software Development. The Center makes these tools available to Federal and State government agencies through its April 1982 Software Tool Catalog, Report No. FCTC-82/013.

In our report on federal agencies' use of software technology, we noted that tools for software testing can reduce the labor of preparing test data and of verifying that test data has caused the logic of the programs to be used.[4] This increases productivity, makes more thorough testing feasible, and results in

---

[3] A software tool is a specially designed computer program that can automate some of the labor involved in the management, design, coding, testing, inspection, or maintenance of other programs.

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

more reliable software. However, we also noted in this report that software tools, including those for testing, were not used consistently throughout the federal government.

At least one agency we reviewed required that software tools be used in the testing process. The agency requires that its new programs be run against a test coverage analyzer called the COBOL (Common Business Oriented Language) Instrumentation Package. However, our review at other agencies and our survey results indicated that most installations do not use software tools for testing. Only 13 percent of the survey installations said they used software test tools.

One reason for this limited use appears to be a lack of training. Our survey showed that only 24 percent of the installations had offered training in the use of tools for software testing. The greatest need for such training is in larger installations: 62 percent of installations with more than 50 employees indicated a great need for additional training. The table below summarizes and projects the responses of all the installations.

	Number of Staff (note a)							
	Fewer than 10		10 to 31		31 to 50		Over 50	
Need	No.	%	No.	%	No.	%	No.	%
Great	235	23	88	34	37	50	96	62
Moderate	221	21	111	43	22	30	29	19
Some	243	23	29	11	7	9	7	5
Little to no need	302	29	22	9	7	9	15	10
No answer	38	4	8	3	1	1	8	5
Total	b/1,038	100	258	100	74	b/100	155	b/100

a/ Based on a projected total of 1,526 installations with at least one business application program.

b/ Total does not add due to rounding and weighting of data.

Installations also indicated a need for training in software testing techniques (walkthroughs, inspections, and so forth). Training in these and other techniques can improve testing productivity and effectiveness. About 65 percent of the installations reported a need for additional training in testing techniques. Again, this was particularly true for the larger installations.

In addition to developing software tools, the Federal Software Testing Center provides support to agencies by installing such

tools and teaching the technology necessary to use them effectively. The Center

--researches the availability of software tools, technology, and services that are available from the private sector and identifies these sources for agencies in need of them,

--develops techniques and procedures regarding the validation of software and quality control measures, and

--endorses newly developed techniques for testing and validating software.

Another part of GSA's Office of Software Development--the Software Testing Branch--also helps agencies in the use of software tools by developing testing guidance to be followed when using these tools for quality assurance.

SOFTWARE TESTING HELPS ENSURE ADEQUATE INTERNAL CONTROL SYSTEMS

Federal legislation requires that agencies establish and maintain adequate systems of internal control. Software testing helps ensure the adequacy of internal control systems by determining whether software controls are adequate and working properly. Thus, software testing helps federal managers comply with statutory and policy requirements for effective internal control systems.

The Budget and Accounting Procedures Act of 1950 requires the head of each federal department and agency to (1) establish adequate accounting and administrative controls to safeguard assets, (2) assure reliable financial information, and (3) assure adherence to applicable laws, regulations, and policies. The Federal Managers' Financial Integrity Act of 1982 (Public Law 97-255), also requires that internal accounting and administrative controls be established. The act further requires heads of federal agencies to prepare annual statements on whether such controls provide reasonable assurances that assets are safeguarded and accounted for accurately and reliably.

By verifying that software does only what it is supposed to do, testing shows that internal controls contained in the software perform as intended. Errors detected in testing may also indicate a need for additional controls. In addition, executive branch policies on computer security specifically require software testing before computer systems containing sensitive information are put into operation.[5] This testing is designed to ensure adequate safeguards and controls in computer systems that have a high potential for loss of assets or sensitive information.

---

[5] OMB Circular A-71, Transmittal Memorandum No. 1, July 27, 1978.

CHAPTER 3

POOR TESTING RESULTS IN COSTLY AND UNRELIABLE SOFTWARE

Better management practices as discussed in chapter 2 could have helped prevent poor testing. Because of poor testing, some agencies did not detect the presence of material errors before software was put into operation. For example, just one error in an operational system required an agency to review thousands of pay accounts and make corrected payments manually. Correcting the program error itself cost $10,000. Had the error been caught during development, before the program was placed in operation, the correction would have cost less. One error in another software program caused an agency to exceed contract progress payments by more than $500,000.

Poor testing had also failed to disclose that software would not do what the user wanted it to do, and as a result the software had to be modified or redone to perform correctly. Such errors reduce the extent to which agencies can rely on their systems to safeguard assets and provide accurate information.

COSTLY ERRORS AND REDUCED SOFTWARE RELIABILITY RESULT FROM POOR TESTING

We found examples of errors in all but one of the 12 business application software systems we examined. In each case the agency could have detected the error through testing before putting the software into operation. These errors, many of which are discussed as case studies in appendix II, required additional funds to correct and disrupted agencies' operations by forcing them to use manual or less efficient methods of processing to accomplish their tasks and correct the errors' effects. On the other hand, we found that the beneficial results of good testing practices were evident in a thoroughly tested system we examined which had few errors and was not costly to maintain.

While software testing cannot be expected to uncover all errors, it should be thorough enough to reasonably assure users that the software will operate accurately and reliably. Detecting all errors would require testing for every possible input condition--an alternative that is neither practical nor feasible. Instead, good testing detects as many errors as possible using representative input data or conditions for which the intended results are known.

It is not easy to develop representative data and conditions that will reasonably ensure software correctness. Those responsible for testing, preferably working with the help of the user, must devise test transactions that include invalid and unexpected conditions as well as valid and expected conditions. In addition,

other factors--such as the uniqueness of the application, the potential cost of an error or malfunction, or the cost of computer and staff resources used in testing--may influence the degree of testing needed.

In the examples discussed below, agencies did not use adequate test transactions and conditions to test the software, and so did not detect material errors. Generally, adequate test cases and conditions were not used for one or more of the following reasons:

--Testing needs for the particular software were poorly analyzed.

--Constraints were placed on time and/or resources.

--The user was not involved in developing the test data and conditions.

Contract administration system overpays progress payments

Even though the agency in case study 1 had tested the software, it had not used test transactions and conditions for payments generated from manual data input. (See p. 29.) Also, apparently because of time pressures, some problems identified during testing were not corrected before the system became operational. As a result, the agency's automated contract administration system made erroneous contract progress payments totaling more than $500,000. The cost to correct these errors included $3,000 worth of programming time, loss of interest on the money overpaid, and additional undetermined costs by regional offices to manually identify the erroneous payments.

At the time of our review, only two of the agency's nine regions had identified erroneous payments caused by the software problems. One identified 11 incidents where contract progress payment limits were exceeded by $500,000 and 2 incidents where such payments were $18,000 too low. Overpayments usually cost the federal government interest on the funds for the period of overpayment.

The second region identified three overpayments. One of these, made in February 1982, overpaid a contract's total price by $43,965. Notified of the overpayment by the agency, the contractor reimbursed this amount in July 1982.

Payroll system makes incorrect housing allowance payments

In case study 2, the agency tested a modification to its payroll subsystem, a variable housing allowance (VHA) system, but did not test for certain pay conditions of the VHA computation for all pay grades. (See p. 31.) Once in operation, a programming error in the system caused a wrong percentage multiplier to be used

for the untested pay grades. The agency estimated that about 750 pay accounts were incorrect. While the costs of correcting this error could not be determined, the effects on payroll processing were disruptive. The payroll processing program aborted before completion and required emergency repairs. Also, the 750 accounts had to be reviewed manually and corrected in a later pay period.

In our review of this case, we noted at least two testing-related problems that contributed to the undetected error. First, and most obvious, the testing criteria for the VHA system did not specify that all pay grades be tested for certain pay conditions, so neither the testing group nor the user developed specific test data for these conditions. Secondly, even though planned tests had not been completed, agency officials directed that the programming changes be made operational. The agency accepted the higher risk of error caused by inadequate testing to avoid delays and nonpayment or to avoid manual payment of the allowance to some 800,000 recipients.

Payroll system computed employees' individual pay in excess of $1 million

In case study 3, the agency designed its payroll system to assume that data already in the data base were complete. (See p. 33.) When this assumption was violated, such as when data were missing in an employee's record, the system computed an erroneous payroll amount for that employee. However, the agency did not test the data base for missing data in employees' records.

The first discovery of such errors was made in early 1979, when the system computed the gross biweekly pay for two employees at more than $2 million each. In response to these errors the agency added a control that would identify excessive payroll calculations for review, but still did not test the data base for missing data. In October 1981, this type of error occurred again when the system computed an employee's gross biweekly pay amount at $1,097,664. The system's new control identified the erroneous pay computation and the agency experienced only minor payroll processing delays. Despite the latest error, the agency did not test for missing data throughout the data base and the potential for future errors still existed.

In March 1982, systems development and payroll/personnel officials, recognizing the potential for errors that the system's controls might not detect, began developing a data base analyzer program. This program, now run biweekly, tests the data base to identify instances of missing data, and thus prevents the errors they could cause.

Omitted and incomplete testing causes payroll errors

In case study 4, the agency failed to test a payroll system modification. (See p. 36.) Originally, programming costs for the

modification amounted to only $57, but because of an undetected error the agency had to correct the program code and the affected master pay files at a cost of more than $10,000. Agency officials estimated that the error had the potential to miscalculate 130,000 pay accounts and to improperly pay 10 percent of these. Agency field offices actually had to review about 5,000 pay accounts and make manual corrections.

Unit and system tests were not performed for this modification because the programmer and the system test office considered the change minor and because of a rush to process the December 1980 payroll. An independent test group responsible for verifying that the modification was correct did perform limited testing, but did not develop adequate test data and conditions. As a result, testing did not detect the programming error--a missing period (.) at the end of a line of code.

Proper testing yields trouble-free system

Besides the examples of poor software testing discussed above, we also noted an example where thorough software testing contributed to a relatively trouble-free system. Begun in 1979, this project was to redesign a subsystem of the agency's payroll system. Representatives of the agency's systems development staff and the user conducted a detailed walkthrough to review and modify the initial functional systems requirements developed by a contractor. In March 1980, this staff issued an implementation plan which included (1) developing test data for unit and systems tests, (2) preparing test plans, and (3) executing and evaluating tests. According to a project team leader, the implementation plan was executed essentially as written. In December 1980, the user accepted the subsystem after extensive systems, parallel, and user acceptance testing.

We examined the subsystem, which consists of about 30,000 lines of COBOL code in 15 program modules. Since implementation, it has operated without problems, and in fiscal 1981, it required less than 200 staff hours of maintenance work--a very small amount. Officials at the facility attribute the subsystem's quality to management's willingness to commit adequate resources to software testing before placing the subsystem into operation.

Poor testing results in unreliable software

Although the errors in the case studies we have discussed are now identified for correction, these systems may still contain other errors undetected because of poor testing. And those potential additional errors may affect the extent to which agencies can rely on them. For instance, the automated contract administration system may contain additional undetected errors because the agency did not adequately test conditions for progress payments

made using manual data input. Also, additional undetected errors involving certain pay conditions for the pay grades that were not tested could also exist in the variable housing allowance system.

In another example, poor software management and testing practices raised questions about the reliability of computer software even though we were not aware of any significant errors detected in this software so far. (See p. 38) The agency we visited has developed a computer model to make actuarial valuations of the civil service retirement plan. As we discussed in an October 1982 report, the reliability of these programs is questionable, however, because the programs were not independently tested and modifications were not controlled.[1]

The model consists of a series of computer programs, which are an integral part of the valuation process. The agency uses data produced by these programs in the retirement fund financial statements and in determining the level of federal funding required for the fund. For example, the programs calculated the actuarial present value of future retirement benefits as $814.3 billion at September 30, 1980, and required funding at $10.9 billion for fiscal year 1980. These programs also help in estimating the retirement portion of a fringe benefit factor federal agencies use in considering the relative cost of acquiring products and services from commercial sources versus providing them in-house.

The agency did not establish an adequate system of internal control over the development and modification of the valuation programs. Specific control deficiencies we noted in our report were as follows:

--Computer program documentation was not properly developed and maintained.

--Programs were not independently tested.

--Program modifications were not adequately controlled.

The agency's actuaries both wrote and tested the programs and made subsequent modifications themselves. The programs were not tested independently before becoming operational. Such independent testing helps ensure objectivity and the development of adequate test cases and conditions. Programmers should avoid final testing of their own work because:

--Someone other than the development programmer can be more objective.

--A programmer's misunderstanding of requirements or specifications may be perpetuated in the testing process.

---

[1] "Inadequate Internal Controls Affect Quality and Reliability of the Civil Service Retirement System's Annual Report," (AFMD-83-3, Oct. 22, 1982).

--The programming organization's cost and schedule objectives may be given precedence over adequate testing.

In addition to not independently testing, the actuarial staff did not retain test data or test results, so it had no documentation. After our review the agency hired another actuary who reviewed the valuation programs, and both actuaries will participate in future modifications to the programs and will check each other's work. This procedure does not, however, constitute independent testing for future modifications to the valuation programs.

BETTER TESTING AGAINST USER REQUIREMENTS IS NEEDED TO ENSURE THAT SOFTWARE MEETS USER NEEDS

A critical objective of software testing is verifying that the software satisfies the user's needs. But our review showed that agencies did not always use software testing effectively to verify that the software did what the user wanted. Limited user involvement in the testing process and inadequate test criteria were the testing deficiencies we noted. These deficiencies caused agencies to modify or redesign the software at additional cost to satisfy user needs.

Poorly translated user requirements can lead to the development of software that does not do what the user wants. For this reason, verifying that software development products (for example, requirements, specifications, and program code) meet user needs should be done throughout software development, and software testing is an important technique in this verification process. Some types of testing that can specifically address whether the software development products meet user needs include inspections, walkthroughs, and acceptance tests.

User involvement is essential throughout system development processing, including software testing. Users share the responsibility of making sure the software meets their needs and requirements. In addition to helping define initial functional requirements, users should help develop appropriate test data and conditions, participate in reviews of software products, and participate in acceptance testing.

In case study 6, we found that the final software product did not satisfy user needs and software testing had not disclosed the problem. (See p. 40.) One agency tested and implemented a new, contractor-designed system for retrieving photographs. However, the agency's acceptance testing failed to disclose that response times for on-line inquiries would become unacceptably long as the data base became fully loaded. Instead of a desired average response time of about 15 seconds per inquiry, actual response time was from 40 minutes to an hour with only 125,000 of the full load

of about 150,000 records loaded in the data base. As a result, the agency (1) resorted to batch processing to minimize costs and delays caused by the lengthy response times and (2) is now spending more than $19,000 to shorten response times by redesigning the data base and rewriting program modules.

Original system specifications did not define an acceptable inquiry response time, nor was this a specific acceptance testing requirement. Acceptance testing was done with only 50 records in the data base and failed to consider the impact on response time of a data base. We believe response time is a critical factor in designing an interactive system. Therefore, testing should have been more thorough and should have included the effect on response time of adding more records to the data base.

We noted instances at several agencies of little or no user involvement in the testing process. Our survey of data processing installations also indicated limited user participation in testing. (See p. 48.) For instance, data processing installations responding to the survey indicated that the user provided data for acceptance testing of software developed in-house only 43 percent of the time, and reviewed and validated test results 67 percent of the time. Moreover, these installations indicated that of their total business application programs developed in-house, only about 14 percent had been tested by the user staff.

TESTING REDUCES THE COST OF ERROR CORRECTION

The examples discussed earlier in this chapter show that errors in operational software can be expensive. Adequate testing detects errors in the developmental phase, thus reducing error correction after the software becomes operational. Generally, the costs associated with the errors in operational software that we noted included:

--Additional computer resources and ADP staff time to reprocess data and to correct program code and affected data files.

--User expenses to manually process or review data and correct the computer's errors.

--Losses of financial assets.

Obviously, some costs--such as user expenses and asset losses--can be avoided if the error is corrected before software becomes operational. But certain ADP costs can also be less if errors are corrected before the software is put into operation.

In our examples of software errors, agencies used both data processing staff time and computer time to correct program code or affected data files, to rerun programs or systems, and to test the corrected software. The amounts of these correction costs varied

with the error. For example, one agency used ADP resources valued at about $270 to correct an error in its payroll system (see p. 34), while another agency needed approximately $10,000 in staff and computer time to correct an error in its automated contract administration system (see p. 37).

It is generally accepted that the earlier an error is detected and fixed, the fewer resources it takes to do so. One reason for this is that as software develops from a concept to an operational program, more stages of that development--such as requirements, design specifications, or program code--may be affected by an error. This is particularly true for design errors that involve changes to software specifications. One industry study noted that such errors are more than 7 times more expensive to correct after software becomes operational than if detected during unit testing, and 30 times more expensive than if detected during design.

We agree that correcting errors during software development generally requires less data processing cost than correcting them in operational software. Therefore, more thorough testing could substantially reduce the cost to correct errors. For example, in 1981 one agency used $1.6 million, or 14 percent of an estimated $11.2 million in direct ADP personnel costs, to solve system problems. Considering estimates of the relative cost of error correction during the software life cycle, substantial savings government-wide from additional errors corrected during development could be dramatic.

CHAPTER 4

CONCLUSIONS AND RECOMMENDATIONS

CONCLUSIONS

Federal agencies spend billions of dollars each year to develop and maintain computer programs used for business applications--computer software that processes transactions affecting agency financial and information resources and that provides information for management decisionmaking. However, our review indicated that federal agencies generally are not managing the software testing process effectively to help ensure that software performs its intended functions accurately and reliably. We believe that undetected software errors are costing agencies millions of dollars unnecessarily because such errors (1) cost more to correct after software becomes operational, (2) often require expensive manual processing or other corrective action, and (3) sometimes result in loss of financial assets.

Software testing is often the last opportunity for managers to check the accuracy and reliability of software before it becomes operational. But, given the cost of errors and their potential impact on the ability of the agency to perform its mission, Federal managers do not emphasize testing strongly enough in the software development and modification processes. While most agencies establish some software testing policies and requirements, they are generally not enforced nor are they communicated through formal written guidance to those responsible for testing. Too often decisions on the amount and extent of testing depend on the discretion of individuals or on time and staff limitations, not on testing policies and requirements for the agency. The examples noted in our review show software that received less than prescribed testing, or no testing at all. Systems development managers waived or did not enforce testing requirements, and costly, disruptive errors occurred.

The users of business application software also have a responsibility to help ensure that software performs as intended. However, users do not always participate in the testing process, and user roles and responsibilities in testing are not always clearly defined. This situation contributes to the development of software that does not meet user needs.

Most agencies do not use software problem data to evaluate the overall effectiveness of the testing process in producing quality software. The agencies we reviewed generally did not use data on the type, frequency, or cost of software errors to obtain feedback and improve testing activities. In fact, most Federal ADP installations responding to our survey did not routinely maintain this type of information.

Federal agencies also have not taken advantage of software technology that could improve the testing process. Staff at most ADP installations do not have training in software testing tools and techniques. Such training could improve testing thoroughness and efficiency. GSA's Office of Software Development provides assistance to agencies in acquiring and using software tools and techniques throughout the software life cycle. We believe that office can actively increase Federal agencies' use of modern testing tools and techniques.

RECOMMENDATIONS

We recommend that heads of Federal agencies:

--Establish written software testing policies and requirements defining the testing procedures, criteria, and techniques required before either agency- or contractor-developed software is placed into operation. These should include specific requirements for user participation in the testing process.

--Monitor and enforce compliance with testing policies and requirements.

--Periodically evaluate the software testing process to determine

(1) its effectiveness in preventing errors and reducing costs associated with error correction and

(2) appropriate allocation of staff and computer resources to software testing.

--Identify and incorporate into the testing process those automated tools and testing techniques that can help the agency provide more thorough testing and more efficient resource use. This should include providing appropriate training on these tools and techniques.

We recommend that the Administrator of General Services, through the Office of Software Development, review selected software development projects in Federal agencies to identify uses and potential uses of software tools and techniques that improve testing thoroughness and efficiency. This office should then report on these reviews to provide guidance to agencies for implementing tools and techniques in their testing processes.

AGENCY COMMENTS AND OUR EVALUATION

GSA agreed with our recommendations on the need to improve software testing thoroughness and efficiency in the federal government. The agency has acted to improve software by assisting agencies in accepting and using state-of-the-art software

technology. We endorse GSA's efforts in assisting agencies as an initial step. However, much remains to be done to improve testing in view of the billions of dollars spent annually on software.

AGENCIES AND INSTALLATIONS VISITED AND SURVEYED

AGENCY/INSTALLATION	VISITED	SURVEYED
1. U.S. Army Finance Center, Ft. Benjamin Harrison, Indiana	X	
2 U.S. Air Force Logistics Command Wright-Patterson Air Force Base, Ohio.	X	
3. Defense Systems Automation Center Columbus, Ohio.	X	
4. Department of Housing and Urban Development Washington, D.C.	X	
5. Department of the Interior Bureau of Reclamation Sacramento, California	X	
6. U.S. Naval Regional Data Automation Center, Naval Air Station Alameda, California	X	
7. Office of Personnel Management Washington, D.C.	X	
8. U.S. Postal Service, Postal Data Center San Francisco, California	X	
9. ACTION		X
10. Department of Agriculture		X
11. Department of the Treasury Alcohol, Tobacco, and Firearms		X
12. U.S. Army a/		X
13. U.S. Air Force a/		X
14. Bureau of Indian Affairs		X

	AGENCY/INSTALLATION	VISITED	SURVEYED
15.	Bureau of Mines		X
16.	U.S. Customs Service		X
17.	U.S. Coast Guard		X
18.	Defense Contract Administration Services Region		X
19.	Defense Contract Audit Agency		X
20.	Defense Logistics Agency		X
21.	Defense Communications Agency		X
22.	Defense Nuclear Agency		X
23.	Department of Energy		X
24.	Export-Import Bank		X
25.	Federal Aviation Administration		X
26.	Federal Highway Administration		X
27.	Federal Law Enforcement Training Ceter		X
28.	Federal Communication Commission		X
29.	U.S. Fish and Wildlife Service		X
30.	U.S. Geological Survey		X
31.	General Services Administration		X
32.	Internal Revenue Service		X
33.	Department of the Interior a/		X
34.	Department of Justice		X
35.	National Aeronautics and Space Administration		X
36.	U.S. Navy a/		X

AGENCY/INSTALLATION	VISITED	SURVEYED
37. National Institutes of Health		X
38. National Oceanographic and Atmospheric Administration		X
39. Nuclear Science Foundation		X
40. National Science Foundation		X
41. Office of the Special Trade Representative		X
42. Department of State		X
43. Tennessee Valley Authority		X
44. Department of the Treasury		X
45. Veterans Administration		X
TOTAL	8	37

a/Excludes agency/installation visited.

CASE STUDIES

CASE STUDY 1 - Poor testing led to erroneous contractor progress payments and contract overpayment

Problem statement

In this case, inadequate testing procedures during the development and testing of an automated contract administration system caused the system programs to make 11 overpayments of contractor progress payments totaling more than $500,000.

Discussion of problem

In May 1981, the agency's validation and automatic progress payment subsystem calculated overpayments totaling more than $500,000 and two underpayments totaling more than $18,000 in one regional office.

For the same period, another regional office reported at least two known overpayments for an undisclosed amount. In response to our inquiry, the second regional office disclosed another overpayment totaling $43,965 made in February 1982. In each instance, the problem identified was erroneous code in the three programs that were used to calculate limitations for progress payments.

In June 1979, the agency modified its validation and automatic progress payment subsystem, enabling the programs to calculate a limitation for progress payments and record manual invoice payments. The modification was tested in July 1979 to determine whether the system specifications were in compliance with the functional requirements. The test identified 37 types of problems, all related to the progress payment calculation. In August 1979, tests were rerun and it was reported that the outstanding problems were corrected. However, in June and July 1981 the payment limitation programs caused the operating system in two regional offices to fail. We identified the problems as (1) erroneous program code, (2) inadequate test data, (3) inadequate test conditions, and (4) imprecise specifications for manual progress payments. The agency apparently corrected the programs, but as late as February 1982 another overpayment totaling $43,965 was reported.

Apparently, the agency did not adequately test possible conditions for progress payments made by inputting data manually. Instead, it addressed the problems as though they were maintenance activities rather than testing and design activities. For example, an important aspect of the testing process is the use and retention

of test results and analysis. However, agency officials could not provide us with results of the tests conducted in July and August of 1979. Moreover, one official told us that the problem resolution apparently "fell through the cracks" amid the hustle of getting the system tested on time. In addition, both a lead programmer and a system manager acknowledged that programmers tested as little as possible and were not knowledgeable about test procedures and available aids.

Impact of errors

The agency spent less than $3,000 to identify and correct programming errors in the automatic progress payment subsystem. However, the actual dollar impact of these errors is not known. Some instances may still exist in which progress payments were overpaid or underpaid, the net effect of which could have cost the Government lost interest and unrecovered overpayments. The recurrence of this type of error in February 1982 increases the doubt that the system accurately and reliably calculates contractor progress payments.

Two of the agency's nine regional offices attempted to identify possible erroneous progress payments caused by programming errors. One region identified 11 overpayments and 2 underpayments, amounting to net overpayments of more than $500,000. The other region identified at least three overpayments with the latest one--for $43,965--occurring in February 1982. That overpayment actually overpaid a contract, and the agency has sought to recover the $43,965 from the contractor. Other cases like this could exist.

The recurrence of the overpayment problem in February 1982 also indicates that the system still does not accurately calculate contractor progress payments. For this reason, the automatic progress payment subsystem's reliability is questionable. Agency officials cannot be reasonably sure that the subsystem adequately safeguards financial assets or that it contributes to their efficient use.

CASE STUDY 2 - The testing process that entered program changes without being validated

Problem statement

In this case, testing for a new variable housing allowance (VHA) system did not include testing criteria for employees within special pay categories. The lack of criteria caused the wrong percentage to be used in computing the new variable housing allowance for officers in pay grades 04 (major) through 06 (colonel) with 4 years enlisted or warrant officer service. As a result, for 750 pay accounts, the agency had to correct the program error for the new housing allowance, test and analyze the results, and recompute (adjust) each account.

Problem discussion

As early as March 1981, the agency began work on modifying its payroll system to include new programs for the VHA subsystem. Between March 1981 and August 1981, extensive effort was devoted to programming and testing the system changes. Although July 25, 1981, was scheduled as a first system test date for the VHA program changes, the tests could not be conducted then because the programming effort was not completed.

In spite of this, the agency's independent quality testing and validation group tested the VHA program changes on August 22, 1981. Officials said they could not complete their review of the testing outputs because they did not have a 2-month pay history experience factor, as required for testing payroll changes. Higher officials directed the testing group to put the program changes into payroll production, regardless of the lack of validation.

During the first production payroll run in October 1981 the the payroll system, containing 800,000 master records, aborted the program before the system could complete the entire payroll. Four major system failures were identified, two of which involved the VHA subsystem. Certain emergency tasks had to be performed to get the system operating again, leaving the problems to be analyzed and corrected later. Consequently, 750 pay accounts were erroneously processed and provision had to be made in another pay period for retroactive correction.

Initial problem solving of the VHA failures revealed an improper posting of the VHA pay entitlements for certain officer grades with prior enlisted service. Researchers concluded that the VHA problem was caused by incorrect coding, resulting in the use of the wrong percentage multiplier for the computation of allowances for officers in pay grades 04, 05, and 06.

In spite of the agency's monitoring of programming staff work output, no test conditions were created specifically for these pay grades. Also, the quality testing and validation group did not receive the test data and specifications it needed to test and validate for conditions before releasing the system into production.

We believe that poor software test planning and lack of user involvement in specifying test conditions contributed to the implementation of programs which produced erroneous pay entitlements.

We found no specifications for test data or documentation for testing special pay categories, and the systems analyst group treated these special pay categories like other employees in the same pay grades.

Impact of errors

Although agency officials did not know the exact number of officers who received incorrect payments, they estimated 750 pay accounts could have been affected. The errors created an unnecessary workload for ADP and accounting staff because (1) coding program changes and retesting program changes had to be made and (2) field offices had to identify payees and compute payments manually.

CASE STUDY 3 - Inadequate testing for complete and accurate data causes incorrect payroll amounts

Problem statement

In this case, an agency did not provide adequate testing, considering the design of its payroll system, to ensure complete and accurate data in the employee master payroll file. As a result, the integrated payroll system calculated excessive payroll amounts when expected data in the data base were missing. Since becoming operational in 1979, the system has continued to compute biweekly gross pay for some individuals in excess of $1 million. Although internal controls single out such large amounts for review, system development and payroll officials agree that other instances of missing data could cause erroneous payroll computations which the internal controls would not detect. In March 1982, the agency's systems development staff began developing a data base analyzer program which edits the data base for missing data. This analyzer program is now run for each biweekly payroll.

Problem discussion

The agency's integrated personnel and payroll system computed an employee's biweekly gross pay amount as $1,097,664 for the payroll period ending October 17, 1981. The system's internal controls detected the error, which was caused by the computer reading blank spaces, but it threw payroll processing totals exactly $1 million out of balance. The agency had to partially rerun the payroll process before preparing the check issuance tape, which is sent to the Department of the Treasury.

Another system control also identified the excessive pay computation. The system produces a list of all employees whose biweekly gross pay exceeds the maximum biweekly salary payable from the Federal Government's General Schedule. The maximum in effect at the time of this error was $2,211.20. The review list showed the employee with a biweekly gross pay of $97,664.00 and a net pay of $10,968.72 because the system limits the field size for gross pay to hundreds of thousands (six positions to the left of the decimal). Therefore, the individual's pay amount was not $1 million or more, but was large enough to be singled out for review. Fields for totals in the system, however, were large enough to contain the total gross pay computation. With the $1 million figure included in the total payroll amount, but not in the individual's pay amount, the reconciliation showed the payroll processing out of balance by that amount.

At the formal request of the agency's payroll staff, systems development staff identified the cause of the out-of-balance condition. To correct it, an edit was added to a payroll program to

prevent this data field from being used in the pay computation when it contains spaces.

Despite similar past errors, as discussed below, and a lack of edits in payroll programs to indicate blank data fields, the agency did not test the data base to see if there were other instances of missing data. Instead, the agency relied on the internal control that identifies payroll computations in excess of the maximum biweekly gross pay.

The agency's office of inspector general reviewed the reconciliations for the first five biweekly payrolls produced by the personnel/payroll system (pay periods ending Jan. 27 through March 24, 1979). Its October 30, 1979, report identified two instances in which blank data fields caused the system to calculate biweekly gross pay of over $2 million for each of two employees. As corrective action, the agency agreed to modify the system so that it would identify all employees whose pay computation exceeded specific limits. This resulted in the biweekly gross pay review list discussed earlier. We could not determine whether testing prior to system operation contained test cases and conditions for missing data.

The inspector general report noted that the agency designed the system with most internal controls intended to prevent entry of erroneous data. Therefore, the system assumes that all data in the personnel/payroll file are complete and accurate. The lack of edits to detect missing data in many data fields is a good example of this assumption. However, the report also points out, and we agree, that maintaining data files without some errors and omissions is virtually impossible. We believe the latest excessive pay computation best illustrates this.

Impact of error

The measurable cost impact of the October 1981 pay computation error was negligible, but the recurrence of this type of error raises questions about the reliability of payroll processing. Corrective action for such errors, including the addition of certain internal control features, did not adequately address the overall problem of potential missing data throughout the data base. Therefore, given the overall system design assumption of an accurate and complete data base, agency officials could not be sure the system accurately computed payroll amounts.

We measured the direct impact of the October 1981 error using ADP-related costs to determine the error, modify the appropriate payroll program, and rerun the payroll. These costs totaled less than $270. However, we believe that, although the ADP-related costs associated with the error are not substantial, the real

impact of the error is the doubt it raises about system accuracy and reliability. System internal controls (the biweekly gross pay review list) identified the latest error; an erroneous check was not issued; and the payroll to agency employees was not late. However, corrective action for this and earlier errors relating to missing data did not adequately address the overall potential problems throughout the data base. The pay review list identifies only pay computations that exceed the maximum allowable pay, not erroneous computations that fall below this limit. In addition, the edit added for the most recent excessive pay computation addressed only the data field that caused this specific error. The system did not contain specific edits for all potential missing data conditions. As a result of these factors, agency officials could not be reasonably sure that the system produced accurate pay computations.

In March 1982, the systems development staff began developing a data base analyzer program which will test the data base for missing data. This program, now run for each biweekly payroll, will help provide the continuous testing the system needs to help ensure accurate and reliable payroll processing.

CASE STUDY 4 - Failure to test program modifications causes major account problems

Problem statement

In this case, an agency did not test a modification to its payroll system. An undetected programming error caused pay miscalculations and had the potential to miscalculate up to 130,000 pay accounts involving pay as many as 13,000 people.

The agency spent more than $10,000 to correct the program and the pay account errors.

Problem discussion

The agency modified its pay system in late December 1980 so that it would properly process pay when individuals reported leave more than six months after they used it. The modification took about five programmer hours and cost about $57. In making the coding change, the programmer failed to place a period at the end of a code line. Because the change was considered minor and there was a rush to complete the December 1980 payroll computation, neither the program nor the system was tested. The program change was sent to an independent test group for validation; however, the group did not use the functional specifications documentation to establish test transactions and conditions or use a current data base for making the tests.

Between implementation in December 1980 and the accidental discovery of adverse effects in March 1981, the program error caused pay account miscalculations for individuals assigned overseas whose cost-of-living entitlement changed. When it identified the coding error and the pay account miscalculations, the agency took the following corrective actions:

--All finance offices were notified of the problems and instructed to make local payments where necessary.

--Problem pay accounts were identified, the code error and master pay files corrected, and all miscalculated accounts recomputed.

--A list of 5,000 payees was prepared and sent to 100 finance offices for manual review and update.

--Corrections to the master pay file were manually reviewed.

--Pay account recomputations were tested for accuracy.

To prevent recurrence of the problem, the agency also directed that the standard quality validation program be followed.

Software testing could have identified the programming errors. Common testing techniques that can provide an objective review, such as desk checks, structured walkthroughs, or peer reviews, were not used. The agency's system test office also skipped system testing because it considered the change minor and felt it had higher priority work.

The next step in the normal testing process, validation by the independent test group, was also inadequate. The group verified only that the coding modification produced the desired change; it did not test for possible follow-on effects in other segments of the system. The tests the group did make used an out-of-date data base which did not include October 1980 rate changes to cost-of-living allowance tables. Later calculations using the correct table values helped identify miscalculated pay accounts.

We also noted that the agency did not have formal test procedures, guidelines, or criteria. Instead, managers relied on programmer discretion to develop, evaluate, and test computer programs. In this case the programmer chose not to test and did not anticipate the follow-on error because she did not use system specifications or documentation in making the coding change.

Impact of error

The agency used about 180 staff and 16 computer hours (costing more than $10,000) to correct the pay program error and resolve incorrect pay accounts or payments. In addition, about 100 local finance offices were directed to review 5,000 pay accounts and make necessary manual adjustments. The costs of this field action were not available.

CASE STUDY 5 - Lack of independent testing contributes to questionable program accuracy and reliability

Problem statement

In this case, the agency's actuarial office uses computer programs in making actuarial valuations of the civil service retirement plan. However, the agency did not subject these programs and their modifications to independent testing procedures. Because of this lack of appropriate testing and other internal control deficiencies related to these programs, we found the reliability of these programs questionable.

Problem discussion

In 1977, the agency developed a computer model to make actuarial valuations of the civil service retirement plan. The model consists of a series of computer programs which are an integral part of the valuation process. The agency uses data produced by these programs in the retirement fund financial statements, and in determining the level of funding required by the Congress. For example, the actuarial present value of future retirement benefits was calculated at $814.3 billion at September 30, 1980, and the required funding at $10.9 billion for fiscal 1980. This program also helps estimate the retirement portion of the standard fringe benefit factor for retirement and disability. Federal agencies use this factor in considering the cost of acquiring products and services from commercial sources versus providing them in-house.

The agency's actuarial staff wrote the programs that support the valuation process. This staff also tested the programs and now controls their operation and modification. Since the agency's systems development function did not participate in program development, the programs were not subject to its testing icies. Moreover, the actuarial staff considers the valuation program just a "super calculator" and maintains it in a computer development library where they can operate and change it as desired. This differs from other agency programs which are maintained in a computer production library--where internal controls and other procedures restrict access to programs and their modification.

In a review of the retirement fund financial statements for fiscal 1980, we noted the following deficiencies in control of the valuation process:

--Computer program documentation was not properly developed and maintained.

--The programs were not independently tested.

--Program modifications were not adequately controlled.

We found the entire valuation process needs better control to provide the Office of Personnel Management with reasonable assurance of accuracy.

By not following accepted software testing practices, the agency's actuarial staff did not reasonably ensure the adequacy of testing and hence the reliability of the valuation process. A basic principle of software testing is that the programmers and the programming organization should avoid testing their own programs. Self-testing can be ineffective because:

--Someone other than the development programmer can be more objective.

--Programming errors due to programmer misunderstanding of requirements or specifications may be perpetuated in the testing process.

--The organization's cost and schedule objectives may be given precedence over the adequacy of testing.

However, in this case the agency's actuarial staff both developed and tested the programs for the valuation process. No independent tests were made, and the actuarial staff did not document the test criteria used or the results obtained.

In addition to doing the testing, the actuarial staff controls the operation and modification of the valuation programs. This further weakens the reliability of the valuation process. Although it has its own systems development staff and computer system responsibilities, the actuarial staff runs the valuation programs on equipment managed by another organizational unit, which also has a systems development staff. Systems development officials agree that control of the valuation process was overlooked--neither staff had been given that responsibility. The officials also agreed that had agencywide testing guidance existed, the actuarial office might have been more aware of the need for independent testing.

Impact of error

The agency cannot reasonably assure users that the valuation process produces accurate and reliable estimates for computing (1) the present value of civil service retirement benefits and (2) congressional funding amounts based on these estimates. Other uses of the valuation process, such as calculating for the retirement portion of the standard fringe benefit factor, may also be unreliable.

CASE STUDY 6 : The testing process that did not consider user needs and requirements

Problem statement

In this case, testing for a new, interactive, time-dependent system did not consider the actual performance demands once the system became operational. Acceptance testing used too few test records and did not detect that, as the data base was loaded, the system response time would increase from an optimal maximum of 15 seconds to more than 40 minutes per single inquiry. To shorten response time, the agency is now spending about $19,000 to redesign the data base and will spend additional funds rewriting the program modules.

Problem discussion

In March 1980, a contractor began designing an on-line, interactive system that would allow agency personnel to quickly locate and retrieve agency photographs. In January 1981, the agency implemented the completed photo retrieval system in one of its regions and began loading a data base that would eventually hold some 150,000 records. However, about 9 months after system implementation, the user began complaining of lengthy response time for system inquiries. By February 1982, agency officials estimated that the data base contained about 125,000 records and response time per inquiry took from 40 minutes to an hour. Although this response time compares favorably to the several days a manual search would require, optimal response time for an interactive system would be only 15 seconds per inquiry. Therefore, to reduce terminal tie-ups and other unnecessary costs associated with lengthy response times, the region created a temporary procedure to process retrievals in a "batch mode."

In December 1980, the system has been acceptance tested; and based on this testing, the region's certification review board certified the system for implementation. However, the region arbitrarily selected only 50 test records for acceptance testing and did not consider inquiry response time for a fully loaded data base. In addition, since original system specifications did not include criteria for response time, acceptance testing did not disclose the potential for unacceptable response time once the system was operational.

We believe poor planning for software testing and acceptance contributed to implementation of a system that did not meet user needs. In fact, we found no formal test plan for the photo retrieval system.

Impact of error

The photo retrieval system, as designed, does not provide the interactive capability the user needs, and will not be implemented agencywide unless response times are reduced. In its attempt to reduce response times, the agency has developed a two-phased approach. In the first phase the contractor will redesign the data base and write a program to convert the information in the old data base to the new one. Then the system will be tested to see if the redesign has reduced response time. The cost of the first phase will be about $19,000 and will take nearly 1,000 staff hours to complete. If the first phase reduces response time, phase two will begin. The contractor will modify about a third of the existing program modules to further reduce response time. The cost of this second phase has yet to be determined.

SUMMARY OF QUESTIONNAIRE RESULTS

This appendix summarizes the results of our survey of Federal ADP installations. The purpose of the survey was to determine the status of and procedures used for testing and maintenance in the federal sector. The survey was accomplished through a questionnaire sent to approximately 600 ADP centers randomly selected from a listing of 4,423 such installations on file with GSA. The questionnaire was to be completed by the ADP manager or other official in charge of the ADP facility.

We received completed questionnaires from 477 of the 600 installations, for a response rate of 79.5 percent. Of these 477 installations, 207 had at least one business application program.

Detailed information on our sampling methodology, including sampling errors, is contained in appendix IV.

CHARACTERISTICS OF ADP INSTALLATIONS WITH BUSINESS APPLICATION PROGRAMS

The following paragraphs describe the number of installations that use business application programs, the number of programs in use, the sources for these programs, the primary use made of these programs, and the number of ADP staff at each installation that had at least one of these programs.

Number of installations and business application programs

Based on our survey results, we statistically estimate that 1,526, or 34.5 percent, of the 4,423 ADP installations had business application programs. The number of such programs at each installation varied considerably from 1 to 8,000. We estimate the 1,526 ADP installations had approximately 812,000 business application programs at the time of our survey, or an average of about 500 programs per installation.

Source of business application programs

Business application programs could have been developed by the ADP staff at the installation, by an outside contractor, or by some other source such as a federal agency that loaned its program to the installation. Our questionnaire asked for the source of these programs, and the responses are shown in table 1.

TABLE 1

Source	Percentage of programs (note a)
Installation's ADP staff	69
Contractor	10
Other	21
Total	100

a/Based only on those responding to the question. Two percent did not answer.

Primary use of business application programs

Our survey disclosed that more business application programs were used for managing and monitoring the activities of the agency (for example, issuing licenses or monitoring grants) than for any other single function. As shown in table 2, the top three functions were managing and monitoring, accounting, and inventory.

TABLE 2

Primary use	Installations	
	Number	Percent
Management and monitoring	183,828	24
Accounting	150,388	20
Inventory	130,126	17
Personnel	91,113	12
Payroll	46,964	6
Staff accounting	22,951	3
Other	134,271	18
Total	759,641	100

Size of ADP staff at installations with business application programs

Our survey showed that the size of the ADP staff[1] at the installations varied considerably from one or a few, to as many as 753. The average ADP staff was 29 employees. As shown in table 3, for purposes of our analysis we divided the installations into four categories based on the size of their ADP staff.

TABLE 3

Category	Installations Number	Installations Percent
Fewer than 10 employees	1,039	68
10 to 30 employees	258	17
31 to 50 employees	74	5
Over 50 employees	155	10
Total	a/ 1,525	100

a/Total does not add due to rounding and weighting of data.

MAINTENANCE AND TESTING OF BUSINESS APPLICATION PROGRAMS

To determine the status of the maintenance and testing performed on business application programs, we asked for the following information on their business application software: (1) the amount of time devoted to maintenance and testing, (2) the types of maintenance performed, (3) the percentage of programs receiving acceptance testing, (4) the difference, if any, between contractor and "in-house" ADP staff in performing tests, (5) the resources available for testing, (6) the time users allotted for testing, (7) the amount of test documentation kept, and (8) the amount of other documentation kept, on business application software.

Time devoted to maintenance and testing

According to our survey, the amount of time spent on maintenance was directly related to the number of programs an installation had: the more programs, the more time devoted to maintenance. For example, about 57 percent of those installations with less than 20 programs reported they spent 10 percent or less of their time maintaining them. However, about 50 percent of those installations with more than 550 programs reported spending from 21 to 50 percent of their time maintaining them.

[1]"ADP staff" refers to the full-time equivalent number of computer specialists, computer system analysts, and computer programmers (any series 334 or the equivalent).

The same relationship did not hold for testing programs. No matter how many programs they had, most installations reported spending less than 20 percent of their time testing programs. Tables 4 and 5 show the results of our survey.

TABLE 4

Staff Time Devoted to Maintenance

Percentage of time	Number of installations (note a)							
	Fewer than 20 programs		20-150 programs		151-550 programs		Over 550 programs	
	No.	%	No.	%	No.	%	No.	%
10 or less	148	57	111	27	81	27	37	10
11 to 20	15	6	74	18	15	5	59	17
21 to 50	96	37	177	44	133	44	177	50
51 to 100	0	-	44	11	74	25	81	23
Total (note b)	258	100	405	100	302	100	354	100

a/Does not include installations that did not provide this information.

b/Totals may not add due to rounding and weighting of data.

TABLE 5

Staff Time Devoted to Testing

Percentage of time	Number of installations (note a)							
	Fewer than 20 programs		20-150 programs		151-550 programs		Over 550 programs	
	No.	%	No.	%	No.	%	No.	%
10 or less	155	64	191	47	177	57	155	46
11 to 20	7	3	125	31	66	21	103	30
21 to 50	74	30	74	18	52	17	66	20
51 to 100	7	3	15	4	15	5	15	4
Total (note b)	243	100	405	100	310	100	339	100

a/Does not include installations that did not provide this information.

b/Totals may not add due to rounding and weighting of data.

Types of maintenance

Next we sought to determine what types of maintenance, if any, were required at least once during fiscal year 1981. About 95 percent of the 1,526 installations required some form of maintenance on a portion of their business application programs. No one type of maintenance was performed substantially more times than any other. Table 6 shows the overall percentage for each type of maintenance that the projected 812,171 programs required.

TABLE 6

Types of maintenance	Percentage of programs
Removing defects in program	15
Keeping tables and codes current	14
Enhancing the program beyond the original design objectives	21
Upgrading hardware or software	16
Changing the program because of legislation and/or regulations	10
Other	2

Acceptance testing of programs implemented in fiscal 1981

Acceptance testing of programs involves comprehensively testing the total system software against the system specifications in a operational environment.

We sought to determine how many of the 100,480 business application programs implemented in fiscal year 1981 had been acceptance tested. We estimate that about 80,000, or 79 percent of the programs, had been developed either by contractors or by the in-house ADP staff--8 percent and 71 percent, respectively. Seventy-three percent[2] of the contractor-developed programs were reported to have been acceptance tested before production. Eighty-six percent[3] of the programs developed in-house were reported to have

[2] Based on 8,152 valid cases. This means the number of contractor-developed programs is greater than or equal to the number of programs that were acceptance tested.

[3] Based on 67,339 valid cases. This means the number of programs developed in-house is greater than or equal to the number of programs that were acceptance tested.

been acceptance tested before implementation. This meant that an average of 85 percent of all business application programs had been acceptance tested, with contractors doing more than in-house ADP staff.

Contractor-developed programs vs. programs developed in-house

Next we asked who performed the acceptance testing. Ideally, it should be done by someone other than the developer of the program. However, as detailed in table 7, about half the contractor programs and about three-quarters of the in-house ADP staff's programs were developed and tested by the same person(s).

TABLE 7

Acceptance Testing of Programs

Developed by contractor staff	Percentage tested (note a)	Developed by in-house ADP staff	Percentage tested (note b)
Tested by staff of contractor who developed the program	52	Tested by in-house ADP staff responsible for program development	74
Tested by staff of contractor other than program developer	8	Tested by other in-house ADP staff	10
Tested by in-house ADP staff	28	Tested by contractor staff	0
Tested by user's staff	9	Tested by user's staff	14
Tested by other(s)	3	Tested by other(s)	3
Total	100	Total	100

a/ Based on 5,137 programs.

b/ Based on 40,447 programs. Percentages rounded to 100.

Next, we asked if either the contractor or the in-house ADP staff had performed certain steps in their acceptance-testing process. Table 8 shows the results.

TABLE 8

Steps	Performed by contractor	Performed in-house
	---------(percent)---------	
Written diary of test results kept	25	22
Test results reviewed and validated by user	91	67
Test results formally accepted in writing by in-house ADP staff and/or user	82	42
User-supplied data used in testing	49	43

Resources available for testing business application programs

Because we thought the testing of business application programs could be influenced by the resources at the installation, we sought to determine the adequacy of computer time, ADP staff, and funds. Table 9 gives the results of our survey.

TABLE 9

Response	Accessibility of computer time		Number of ADP staff		Availability of funds	
	No.	%	No.	%	No.	%
More than adequate	943	62	272	18	309	20
Adequate	405	27	590	39	744	49
Less than adequate	147	10	619	41	383	25
No answer	29	2	44	3	88	6
Total (note a)	1,526	100	1,526	100	1,526	100

a/May not add due to rounding and weighting of data.

These statistics imply that the computer is usually accessible and does not hinder testing of business application programs. However, the number of ADP staff available often limits the amount of testing performed. Only 25 percent of installations said funding is a limiting factor.

Time allotted by users for testing business application programs

Next, we asked if the amount of time allotted by the users for testing business application programs influences the testing process. The responses are summarized in table 10.

TABLE 10

Time allotted	Installations Number	Installations Percent
More than needed	44	3
About right	980	64
Less than needed	442	29
No answer	59	4
Total	a/ 1,526	100

a/Total does not add due to rounding and weighting of data.

Test documentation

We asked whether installations had their programs routinely documented according to FIPS PUB 38. This National Bureau of Standards publication recommends that a test plan and a test analysis report be prepared. The test plan should identify test milestones and provide the schedule and requirements. It should include specifications, descriptions, and procedures for all tests, as well as test data reduction and evaluation criteria. The test analysis report should document the test results and findings. That report should also include a summary of the software's capabilities, deficiencies, and recommendations. As table 11 shows, only a projected 185 installations, or about 11 percent, did such documentation routinely in fiscal 1981.

TABLE 11

Amount of progress with documentation	Installations Number	Installations Percent
All or almost all programs	111	7
Most programs	52	3
About half the programs	22	1
Some programs	118	8
Few or no programs	1,105	73
No answer	118	8
Total	1,526	100

Data maintained on fiscal 1981 business application software

Our analysis showed that only a small percentage of the installations maintained any one type of data relative to failures and costs in fiscal 1981. Table 12 shows how many installations maintained five types of data in fiscal 1981.

TABLE 12

Type of data	Installation response					
	Maintained		Not maintained		No answer	
	No.	%	No.	%	No.	%
Software development costs by system	442	29	1,039	68	44	3
Number of lines of code of production software in use	516	34	958	63	52	3
Types of software program failures	472	31	1,010	66	44	3
Frequency of software program failures	450	30	1,039	68	37	2
Cost to repair failures	170	11	1,297	85	59	4

GUIDELINES AND POLICIES

We sought to determine how many installations had standards to help their in-house ADP staff test business application programs, and what standards applied when the contractors tested such programs. We checked to see if written guidance had been received from outside sources such as NBS, GSA, or the parent agency and, if it had, how useful this guidance had been. We asked if additional guidance from any of these sources was needed. We also sought to determine the policy for formal training; that is, how many people had received such training in testing business application programs and whether a need existed for formal training.

Standards for in-house ADP staff

Slightly more than half the installations had no written guidelines or policies to help the in-house staff test business application programs. (See table 13.) Thus, the standards for testing a program could range from very thorough to superficial, depending on the preference of the staff. And since, as discussed earlier, most business application programs are developed in-house and are usually tested by the same staff that developed them, we see potential problems. The in-house staff should at least be given minimum standards to follow when testing programs.

TABLE 13

Category of response	Installations	
	Number	Percent
Have standards	707	46
Have no standards	803	53
No answer	15	1
Total	1,525	100

Standards for contractors

We asked whether contractors received guidance from the installations when performing the testing. Most installations said they provided testing standards for contractors to follow; only 15, or 3 percent, of the installations said they provided no standards. Table 14 gives a breakdown of the standards contractors used--in-house or contractor. No answer was given to this question by 162 installations, and 914 said the question was not applicable to them--they do not use contractors to perform testing.

TABLE 14

Standards used for testing by contractors	Installations	
	Number	Percent
In-house standards	206	46
Contractor standards	29	7
Both in-house and contractor standards	199	44
No standards required	15	3
Total	449	100

Sources of written guidance on testing of business application programs

Next, we asked if in the past these installations had received any written guidance on the testing of business application programs, either from central Government agencies such as NBS or GSA, or from their own department or agency. We also asked how useful any such guidance had been. Responses showed that 760 installations had received written guidance from either NBS or GSA and 678 had received it from their own department or agency. The central guidance was judged to be not substantially useful, while that from individual departments and agencies was judged to be useful overall.

Written guidance from NBS and GSA

Only about 39 percent, or an estimated 590 installations, had received written guidance from NBS. (See table 15.) Table 16 shows how useful the installations considered that written guidance to be.

TABLE 15

Category of response	Installations Number	Installations Percent
Had not received guidance	737	48
Received guidance from NBS	590	39
No answer	199	13
Total	1,526	100

TABLE 16

Degree of usefulness of NBS guidance	Installations Number	Installations Percent
Very great	22	4
Great	37	6
Moderate	192	33
Some	206	35
Little or no	111	19
No answer	22	4
Total	590	100

Even fewer--about 11 percent, or 170 installations--had received written guidance from GSA's Office of Software Development. (See table 17.) Table 18 shows how useful that guidance was considered to be.

TABLE 17

Category of response	Installations Number	Installations Percent
Had not received guidance	1,091	72
Received guidance from GSA	170	11
No answer	265	17
Total	1,526	100

TABLE 18

Degree of usefulness of GSA guidance	Installations Number	Installations Percent
Very great	15	9
Great	-	-
Moderate	29	17
Some	66	39
Little or no	29	17
No answer	29	17
Total (note a)	170	100

a/Totals do not add due to rounding and weighting of data.

Written guidance from department or agency

An estimated 44 percent, or 678 installations, had received some written guidance from their department or agency on the testing of business application programs. (See table 19.) Overall, the installations that had received such guidance thought it to be of substantial use. (See table 20.)

TABLE 19

Category of response	Installations Number	Installations Percent
Had not received guidance	693	45
Received guidance from own agency	678	44
No answer	155	10
Total	1,526	a/ 100

a/Total does not add due to rounding and weighting of data.

TABLE 20

Degree of usefulness of parent agency guidance	Installations Number	Installations Percent
Very great	162	24
Great	162	24
Moderate	221	33
Some	74	11
Little or no	15	2
No answer	44	7
Total	678	a/ 100

a/Total does not add due to rounding and weighting of data.

Need for additional written guidance on testing of business application programs

Only about one-fourth, or an estimated 398 installations, thought there was a need for additional written guidance on the testing of business application programs. When asked who should be the primary source of such guidance, about a third chose NBS. Of these, well over half had received written guidance from NBS in the past and most had found the guidance useful.

Another fourth of the 398 installations thought GSA's Office of Software Development should have primary responsibility for developing this additional written guidance. Most of these installations had not received any written guidance from that office in the past.

We listed one other central Government agency as a possible choice for developing additional written guidance on testing: the Office of Management and Budget. Table 21 shows which of the sources of guidance the installations preferred.

TABLE 21

Organization to have primary responsibility for guidance	Installations' choice Number	Percent
National Bureau of Standards	147	37
GSA's Office of Software Development	96	24
Office of Management and Budget	22	6
Other	111	28
No answer	22	6
Total	398	a/ 100

a/Total does not add due to rounding and weighting of data.

Formal training

We believe that the amount of formal training each ADP staff receives in either testing techniques or in using automated tools could affect the adequacy of the testing of business application programs. With that in mind, we asked what percentage of the ADP staff had received formal training, and whether it was enough. We defined the use of automated tools as (1) program instrumentation, (2) data base extraction or generation, and (3) file comparisons in testing. Our survey showed that 13 percent, or 199 out of 1,526 installations, use automated tools in the testing process. ADP staff at about 24 percent of these had received formal training in the use of automated tools, and on the average, about 30 percent of ADP staff at each installation had formal training in testing techniques.

Was this enough formal training in these areas? According to our survey, the need for additional training in testing techniques varied according to the size of the ADP staff at the installation. As shown in table 22, the larger the staff the greater the need for additional training.

TABLE 22

Need	Size of staff: Less than 10 employees No.	%	10-30 employees No.	%	31-50 employees No.	%	Over 50 employees No.	%
Great	236	23	74	29	22	30	66	43
Moderate	24	2	96	37	44	59	59	38
Some	273	26	59	23	7	9	29	19
Little to no need	287	28	29	11	-	-	-	-
No answer	219	21	-	-	1	1	1	1
Total	1,039	100	258	100	74	a/100	155	a/100

a/Total does not add due to rounding and weighting of data.

We found the same results when we asked about training for tools--the larger the ADP staff the greater the need for additional training in the use of automated tools. The results of our survey are shown in table 23.

TABLE 23

Need	Size of Staff: Less than 10 employees No.	%	10-30 employees No.	%	31-50 employees No.	%	Over 50 employees No.	%
Great	235	23	88	34	37	50	96	62
Moderate	221	21	111	43	22	30	29	19
Some	243	23	29	11	7	9	7	5
Little to no need	302	29	22	9	7	9	15	10
No answer	38	4	8	3	1	1	8	5
Total	1,039	100	258	100	74	100	155	a/100

a/Total does not add due to rounding and weighting of data.

SAMPLING METHODOLOGY, DATA COLLECTION, QUALITY CONTROL, AND PROJECTED RESULTS

This appendix describes how we statistically sampled Federal ADP installations, designed a questionnaire to be sent to these installations, maintained quality control over the data thus obtained, and made projections to the universe.

SAMPLING METHODOLOGY

The General Services Administration (GSA) had 4,423 Federal ADP installations on file. By statistically sampling these installations, we could examine a smaller group of 600 installations (the sample) and then draw conclusions and generalize about all 4,423 installations (the universe).

The results from this or any other statistical sample are always subject to some uncertainty or sampling error because only a portion of the universe has been selected for analysis. The sampling error consists of two parts: confidence level and range. The confidence level indicates the degree of confidence that can be placed in the projections derived from the sample. The range is within the upper and lower limit of responses andcontains the actual universe value. For example, ourstatistical sample showed that 207 installations had business application programs. Using the sampling error formula, we are 95-percent confident that the true number of installations with business application programs would be between 1,370 and 1,682 (or within a range of 1,526 installations ± 156).

DATA COLLECTION

A questionnaire was developed to record information about the testing and maintenance of business application programs. This questionnaire was then pretested to determine: (1) if the target group (managers of Federal ADP installations) possessed the information desired, (2) if the questionnaire would be burdensome on the respondent, and (3) if the questionnaire design--including the print size, the layout complexity, and procedures for recording information--was appropriate. Once it had been pretested and the necessary changes made in its design, the questionnaire was mailed to our sample installations.

QUALITY CONTROL

Maintaining quality control over the data was important. The completed questionnaires were reviewed by the project manager and staff for completeness and accuracy. The data was then keypunched to create a computerized data base. An appropriate sample of this data base was verified with the questionnaires. Computerized logic checks were run to look for incorrect data, and any errors detected were corrected.

PROJECTED RESULTS

After the data base was verified, it was weighted to project the sample results to the universe. The weight was calculated by dividing the universe size by the the sample size (4,423/600=7.37). That is, any condition in one of the 600 installations can be projected to 7.37 installations in the universe.

The following tables show the sampling errors for the projections found in our report.

TABLE 1

Questionnaire Responses

Questionnaire	Sample	Universe estimate Number	Universe estimate Percent	Range (95% Confidence Number	Range (95% Confidence Percent
Returned	477	3,516	79.5	±133	±3
Undelivered	11	81	1.8	± 44	±1
Not returned	112	826	18.7	±128	±3
TOTAL	600	4,423	100.0		

TABLE 2

Number of Installations with Business Application Programs

	Projection to adjusted universe Number	Projection to adjusted universe Percent	Estimated ranges of adjusted universe at the 95-percent level of confidence Number	Estimated ranges of adjusted universe at the 95-percent level of confidence Percent
Installations with business application programs	1,526	34	±156	±4
Other installations	1,990	45	±164	±4
No answer	907	21	±133	±3
TOTAL	4,423	100		

TABLE 3

Characteristics of These Business Application Programs And Their Installations

Sources of business application programs:	Projection to adjusted universe		Estimated ranges of adjusted universe at the 95-percent level of confidence	
	Number	Percent	Number	Percent
Installation's ADP staff	541,352	69	±157,521	±20
Contractor	78,251	10	± 44,933	± 6
Other	168,889	21	± 64,805	± 8
TOTAL (note a)	788,492	100		
Primary use of business application programs:				
Management and monitoring of activities under the various programs the agency is authorized to administer	183,828	24	± 65,195	±9
Accounting	150,388	20	± 43,531	±6
Inventory	130,126	17	± 65,728	±9
Personnel	91,113	12	± 37,463	±5
Payroll	46,964	6	± 12,194	±2
Staff accounting	22,951	3	± 8,236	±1
Other	134,271	18	± 60,132	±8
TOTAL (note b)	759,641	100		

a/Total based on only those who provided valid answers. Five of the 207, or about 2 percent, did not provide valid responses.

b/Totals based only on those who provided valid answers. Nine of the 207, or about 4 percent, did not provide valid responses.

TABLE 3 (cont.)

Size of ADP staff:	Projection to adjusted universe		Estimated ranges of adjusted universe at the 95-percent level of confidence	
	Number	Percent	Number	Percent
Fewer than 10 employees	1,039	68	±140	±9
10 to 30 employees	258	17	± 77	±5
31 to 50 employees	74	5	± 42	±3
Over 50 employees	155	10	± 60	±4
TOTAL	1,526	100		
Amount of time devoted to maintenance:				
10 percent or less	376	28.5	± 92	±7
11 to 20 percent	162	12.3	± 62	±5
21 to 50 percent	582	44.1	±111	±8
51 to 100 percent	199	15.1	± 68	±5
Unknown	206		± 69	-
TOTAL	a/1,526	100.0		
Amount of time devoted to testing:				
10 percent or less	678	52.3	±119	±9
11 to 20 percent	302	23.3	± 83	±6
21 to 50 percent	265	20.4	± 78	±6
51 to 100 percent	52	4.0	± 35	±3
Unknown	229	-	± 73	-
TOTAL	1,526	100.0		

a/ Total does not add due to rounding and weighting of data.

TABLE 4

Programs Requiring Maintenance At Least Once

Types of maintenance	Projection to adjusted universe		Estimated ranges of adjusted universe at the 95-percent level of confidence	
	Number	Percent (note a)	Number	Percent
Changes to remove defects in program	119,599	15	± 69,190	±9
Changes to keep tables and codes current	113,566	14	± 38,571	±5
Changes to enhance the program beyond the original design objectives	173,086	21	± 59,490	±7
Changes due to upgraded hardware or software	133,840	16	± 58,200	±7
Changes due to legislation and/or regulations	82,136	10	± 27,724	±3
Other	13,552	2	± 17,644	±2

a/Percentages are based on 812,171 estimated programs at the installations. Percentages are not cumulative since each program could have been listed under each type of maintenance.

TABLE 5
Resources Available for Testing
Business Application Programs

	Projection to adjusted universe		Estimated ranges of adjusted universe at the 95-percent level of confidence	
	Number	Percent	Number	Percent
Computer time:				
More than adequate	944	62	±135	±9
Adequate	405	27	± 95	±6
Less than adequate	147	10	± 59	±4
No answer	29	2	± 27	±2
TOTAL	a/ 1,526	a/ 100		
Staff:				
More than adequate	273	18	± 79	±5
Adequate	590	39	±112	±7
Less than adequate	619	41	±114	±7
No answer	44	3	± 33	±2
TOTAL	1,526	a/ 100		
Funds:				
More than adequate	310	20	± 84	±6
Adequate	744	49	± 123	±8
Less than adequate	383	25	± 93	±6
No answer	88	6	± 46	±3
TOTAL	1,526	100		
Time allotted by users for testing business application programs:				
More than needed	44	3	± 33	± 2
About right	980	64	± 137	± 9
Less than needed	442	29	± 99	± 7
No answer	59	4	± 38	± 3
TOTAL	a/ 1,526	100		

a/Total does not add due to rounding and weighting of data.

TABLE 6
Resources Available for Testing
Business Application Programs

	Projection to adjusted universe		Estimated ranges of adjusted universe at the 95-percent level of confidence	
	Number	Percent	Number	Percent
Computer time:				
More than adequate	944	62	±135	±9
Adequate	405	27	± 95	±6
Less than adequate	147	10	± 59	±4
No answer	29	2	± 27	±2
TOTAL	a/ 1,526	a/ 100		
Staff:				
More than adequate	273	18	± 79	±5
Adequate	590	39	±112	±7
Less than adequate	619	41	±114	±7
No answer	44	3	± 33	±2
TOTAL	1,526	a/ 100		
Funds:				
More than adequate	310	20	± 84	±6
Adequate	744	49	± 123	±8
Less than adequate	383	25	± 93	±6
No answer	88	6	± 46	±3
TOTAL	1,526	100		
Time allotted by users for testing business application programs:				
More than needed	44	3	± 33	± 2
About right	980	64	± 137	± 9
Less than needed	442	29	± 99	± 7
No answer	59	4	± 38	± 3
TOTAL	a/ 1,526	100		

a/Total does not add due to rounding and weighting of data.

General
Services
Administration Washington, DC 20405

SEP 14 1983

Honorable Charles A. Bowsher
Comptroller General of the United States
U.S. General Accounting Office
Washington, D.C. 20548

Dear Mr. Bowsher:

We have reviewed the draft General Accounting Office report, "Greater Emphasis on Testing Needed to Make Computer Software More Reliable and Less Costly" (GAO/IMTEC-83-3). The General Services Administration (GSA) is in total agreement with the report's findings and recommendations.

GSA established the Office of Software Development and Information Technology to provide assistance to agencies in accepting and using state-of-the-art software technology. To accomplish this, we have been assisting agencies in developing software quality assurance programs that make use of software tools and ensure the acquisition of software that has been adequately tested to meet requirements and standards. We have also been providing training on software tools methodology, selection, and installation.

We periodically document these assistance projects in reports that are aimed at providing other agencies with information on ways in which their ADP costs can be reduced through software testing and the use of software tools. Such reports have included: "A Software Tools Project: A Means of Capturing Technology and Improving Engineering" (OSD-82-101), "Establishing a Software Engineering Technology" (OSD/FSTC-83/014), and "Software Tools Survey" (OSD/FSTC-83/015).

In summary, we feel that improved software testing can significantly reduce Federal ADP costs. We at GSA will continue our efforts to help agencies realize these savings.

Sincerely,

Ray Kline
Deputy Administrator

(913667)

26801

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