Issue Date: March 5, 1996

Report Number: 96-DP-166-0001

Report Title: Controls Over Software Maintenance Must Be Significantly Strengthened

MEMORANDUM FOR: Dwight P. Robinson, Deputy Secretary, SD
Marilynn A. Davis, Assistant Secretary for Administration, A
Andrew M. Cuomo, Assistant Secretary for Community Planning and Development, D
Nicolas P. Retsinas, Assistant Secretary for Housing-Federal Housing Commissioner, H
Kevin E. Marchman, Acting Assistant Secretary for Public and Indian Housing, P
Michael A. Stegman, Assistant Secretary for Policy Development and Research, R
Elizabeth K. Julian, Acting Deputy Assistant Secretary for Policy and Initiatives, EP
John A. Knubel, Chief Financial Officer, F
William E. Dobrzykowski, Vice President, Government National Mortgage Association, TF
Steven M. Yohai, Acting Director, Office of Information Technology, AMI
Craig E. Durkin, Director, Office of Procurement and Contracts, ARC

FROM: Benjamin K. Hsiao, Director, Information Systems Audit Division, GAA

SUBJECT: Review of HUD's Management of Software Maintenance

We have completed a review of HUD's Management of Software Maintenance. Our objectives were to evaluate HUD's:

- Software maintenance policies and procedures;
- Management of software maintenance during the system lifecycle;

- Management of contractors' performance of software maintenance; and

- Management of costs of application software maintenance.

We analyzed aggregate performance data of six critical and one highly sensitive applications to determine if there were indicators of problems in software maintenance. The focus of the audit recommendations was on effective preventive controls.

Our report contains three findings. The findings disclose that numerous controls are needed in software maintenance. To control costs, HUD needs to adopt a project cost accounting system which can uniformly and consistently accumulate, track, allocate, bill, and report costs to system users; apply a consistent definition of software maintenance for project cost tracking and billing purposes; and require that system managers periodically update cost benefit analyses to determine when systems should be redesigned or replaced.

To control the risk of errors and system failures, the Office of Information Technology (IT) must control software changes. System owners need to (1) establish a centralized board to review, evaluate and approve all requests for change, (2) develop a maintenance schedule, (3) define standard classifications of change, and (4) perform sufficient user testing and acceptance of software changes. Further, performance indicators for project tracking or established measurable quality goals should be established and IT must use purchased software tools for configuration control.

Finally, performance based contracting should be used to control the quality of services and products provided by contractors providing software development and maintenance support. HUD must establish standards to monitor contractor performance to prevent poor quality work that could damage critical data, cause systems to fail, and increase costs beyond what is already spent for correcting problems.

We believe the control weaknesses described in the report merit the attention of agency senior management because they could result in system failures that would prevent HUD from fulfilling its mission. Within 60 days, please give us, for each recommendation made in the report, a status report on: (1) corrective action taken; (2) the proposed corrective action and the date to be completed; or (3) why action is considered unnecessary.

Should you or your staff have any questions, please have them call me at 708-3444 extension 149.

Executive Summary

Software maintenance is defined as those activities required to keep an application system[1] operational and responsive after it is accepted and placed into production. It is a costly, technically challenging, and critically important activity. For the last two years HUD spent over $240 million to operate, develop, and maintain its application systems, which process hundreds of billions of dollars worth of transactions in support of program activities. HUD depends on these systems to perform mission critical functions such as subsidy distribution and asset management.

Our audit objectives were to evaluate management controls over the application software maintenance process and the quality and quantity of cost information. We analyzed aggregate performance data of six critical and one highly sensitive applications to determine if there are indicators of problems in software maintenance. The focus of the audit recommendations was on effective preventive controls.

Summary of Findings

Project Cost Accounting Must Be Used To Control Software Costs

Current industry sources indicate that software maintenance can account for as much as 80 percent of an application system's lifecycle costs. These costs must be identified so management can make informed decisions regarding system enhancements or replacements. However, HUD has significantly understated its true software maintenance costs. HUD does not have a project cost accounting system which can uniformly and consistently accumulate, track, allocate, bill, and report costs to system users as intended by OMB Circular A-109.

We reviewed the costs of seven application systems over an 18-month period. Out of a total of $40.8 million billed to users under the Working Capital Fund, $4.7 million of the maintenance costs were classified as development[2] costs that the Federal Information Processing Standards Publication (FIPS PUB) 106, Guideline on Software Maintenance, would have classified as perfective and/or adaptive maintenance. We also noted that $3.18 million in system engineering hours were not consistently treated as either development or maintenance between the project management and cost allocation/billing processes. In addition, maintenance work associated with $29.4 million in computer usage and telecommunications costs were not separately identified. Further, HUD managers were only managing approximately $11.2 million, or 27 percent of the total $40.8 million costs billed for the seven systems reviewed.

There are several reasons why these conditions exist. HUD has not applied a consistent definition of software maintenance for project cost tracking and billing purposes. Also, HUD's current cost allocation and billing process is complex, disjointed, requires much manual intervention, and does not interface with HUD's central accounting system. In addition, the project cost chart of accounts lacks specific accounts of activities necessary for cost measurement purposes. Another reason is that system managers are not required to periodically update cost benefit analyses for major ADP systems.

Without identifying, capturing, and reporting maintenance costs in accordance with FIPS PUB 106, HUD cannot make informed system and budget decisions. In particular, HUD cannot determine when systems should be evaluated for replacement because of excessive maintenance costs. HUD's systems cost more than $110 million a year and at least $1.5 billion over their lifecycle to operate and maintain.

Numerous Controls Needed For Application Software Changes

Controlling software changes is essential to keep HUD's application systems functioning and responsive to user needs. However, despite Federal guidance, we found change control weaknesses in all seven HUD application systems we reviewed. System owners have not (1) established centralized approval and review of changes, (2) developed a maintenance schedule, (3) defined standard classification of changes, and (4) performed sufficient user testing and acceptance of software changes. The Office of Information Technology (IT), in providing technical support, has not formally defined a software maintenance process, established performance indicators for project tracking, or established measurable quality goals. In addition, IT did not use software tools for configuration control for which it has already purchased.

Deficiencies in software change controls have resulted in a significant amount of errors during maintenance. IT's own study showed error correction rates that could be higher than the normal rate defined in Federal Information Processing Standards Publication (FIPS Pub) 106, Guideline on Software Maintenance. We also noted that although 84 percent of the software releases did not result in production problems, 16 percent of all releases created new problems that were very costly in IT and program office staff time. The study also showed that during an eight-month period, 26 releases (a set of software changes) required 44 emergency releases of corrections to software changes that did not work properly after having been tested and approved for release into production. This data and similar results from our analysis strongly suggest that established review and testing

procedures are not adequate, and perhaps contractor performance could be improved. These errors also could indicate that these systems have become unstable or unreliable, and need to be replaced. In addition, these errors could have damaged critical data and caused system failures.

Adequate change controls are not in place because the application systems owners have not been exercising their responsibilities required under Office of Management and Budget (OMB) Circular A-130 to manage their application systems for the successful conduct of the program mission. Another cause is the lack of a software maintenance policy that defines the responsibilities, authorities, functions, and operations of IT and user organizations. A third important reason is the lack of performance measures to aid management in determining the quality of software after the changes are made.

HUD Has Not Been Using Performance-Based Contracting Methods for Software Development and Maintenance Contracts

The Offices of Procurement and Contracts (OPC) and IT have not been using performance-based contracting methods to award software development and maintenance contracts as required by Federal Procurement Regulations. HUD has consistently awarded Cost-Plus-Fixed-Fee (CPFF) contracts rather than using incentive contracts for software development and maintenance. On the eight CPFF contracts providing software support to the seven application systems we reviewed, our evaluation revealed that the Statements of Work (SOWs) lacked measurable performance standards, acceptable quality levels, and quality assurance surveillance plans. Consequently, these eight contracts, worth over $110 million, provide neither the incentive for nor the accountability of contractors to perform well and control costs. Also, without performance standards, HUD cannot monitor contractor performance to prevent poor quality work that could damage critical data, cause systems to fail, and increase costs beyond what is already spent for correcting problems.

Conclusions

We have concluded that HUD does not have effective controls over software maintenance. As examples, HUD cannot make informed decisions on system replacement or redesign because government and industry accepted classifications are not used to capture software maintenance costs. HUD cannot control the quality of software changes because needed data such as number of changes, frequency of changes, modules changed most frequently, failure rate, defects, etc. are not collected and analyzed. HUD cannot hold contractors accountable for the quality and costs of products and services provided because contractor performance

standards have not been established. Finally, HUD lacks a performance measurement program to aid management in determining whether the resources expended for software maintenance are achieving the goal of keeping the application systems functioning and responsive to user needs.

The control deficiencies in software maintenance expose HUD to several high risks. Millions of dollars could be spent each year on application systems that are becoming obsolete and should be replaced. Software changes could be processed without adequate audit trails, and unapproved, unintentional, or malicious modifications could be introduced and proceed undetected through the change process. Errors, system failures, and excessive costs could result from poor performing contractors. The Department should report the control deficiencies in software maintenance as a significant weakness in accordance with HUD Handbook 1840.1, Departmental Management Control Program Handbook.

Recommendations

While we make several specific recommendations, there are two themes running through them--measurement and system owner responsibility. We have established that software maintenance is a costly, technically challenging, and critically important activity. The recommendations on measurements are consistent with General Accounting Office's[3] Best Practice 5, "Measure the performance of key mission delivery processes." The recommendations on system owner responsibility begin to implement Best Practice 9, "Establish customer/supplier relationships between line and information management professionals."

Chief Financial Officer

- Expedite the implementation of the Project Cost Accounting System (PCAS)/Cost Allocation Application; and

- Establish cost accounting standards to ensure maintenance and development project costs are uniformly and consistently accumulated, tracked, allocated, and reported for management and billing purposes.

Acting Director of Information Technology, through the Technology Investment Board (TIB)

- Establish a software maintenance definition that can be uniformly and consistently applied throughout HUD's project accounting and cost allocation and billing processes that meet the intent of FIPS PUB 106;

- Establish appropriate software maintenance

classifications that, at a minimum, include the three categories described in FIPS PUB 106;

- Establish a more measurable software chart of accounts that describes specific development and maintenance activities needed to gain economic insights into the major costs associated with software;

- Establish guidelines and define management responsibilities for capturing all system lifecycle costs to include updating the required cost benefit analyses at appropriate, defined, and periodic intervals;

- Issue guidelines with the concurrence of the Management Committee for system owner organizations to use in managing the software maintenance of their applications in accordance with OMB Circular A-130:

  - Establishing Change Control Boards;

  - Evaluating the systems usability in supporting organizational goals and mission;

- Define software maintenance and classify changes according to the functional type of change being made to the application systems (e.g., adaptive, corrective, perfective);

- Identify the measurements needed to support Departmentwide management of information technology; and

- Adopt a standardized form for initiating all requests for software changes, regardless of anticipated level of effort.

TIB Representatives and System Owner Organizations

- Follow the guidelines issued by the Management Committee to establish a centralized change control board (See Recommendation 3 (a) above);

- Follow the guidelines issued by the Management Committee to periodically evaluate the application systems' capability to support Departmental and office strategic objectives; and

- Follow the guidelines issued by the TIB for conducting user acceptance testing and reviewing the test results of software changes.

Acting Director of Office of Information Technology

- Establish a descriptive maintenance account within the integrated resource accounting system Biller 1100 to capture software maintenance computer usage costs;

- Modify and/or establish policy documents to include an organizationwide software maintenance policy describing in broad terms the responsibilities, authorities, functions, and operations of both system owners and IT;

- Establish a policy for the use integrated software configuration management (SCM) tool, such as ENDEVOR, for all application systems on all platforms (i.e., UNISYS, HDS, LAN, PC);

- Establish policies for software verification and validation (V&V) plans, as defined in FIPS PUB 132;

- Modify the System Development Methodology (SDM) to include the following:

    - Detailed source coding standards applicable to all phases in the SDM;

    - Formal testing procedures for software modifications and acceptance testing by the user community; and

    - An objective review of modified source code and test documentation to ensure the modified code conforms with established source code standards and effectively complies with the change request, and that no additional unapproved changes were introduced by the programmers during the coding process;

- Prepare guidelines to assist IT to develop realistic performance standards that are measurable, including:

    - Resource tracking - quantification in dollar amounts of resources used as the input for production of a service or product;

    - Work product tracking - the number of units of the product or service provided to the customer, and appropriate measures of size and complexity;

    - Quality tracking - the level of service or product quality, both in terms of customer satisfaction (external quality) and of work performed to

provide the service (internal process quality); and

- Change tracking - a common tracking system that will track all software changes made to each application system, regardless of the type of proposed change or its anticipated level of effort.

- Incorporate surveillance plans and performance requirements' summaries into performance-based contracts;

- Develop task specifications that follow internal instructions, IT-CON-04 and IT-CON-05, and include measurable performance standards and acceptance criteria related to each specific task assignment;

- Ensure that modifications to the task specifications provide additional details about the task and/or work requirements;

- Ensure that all product acceptance forms are reviewed timely according to agency standards; and

- Ensure that copies of task specifications and product acceptance forms issued are provided to OPC for their review and evaluation.

Director of the Office of Procurement and Contacts (OPC)

- Implement performance-based contracting methods for software development and maintenance contracts.

Auditee Comments and OIG Response

We provided the draft report to all the HUD program offices on July 12, 1995. We received extensive verbal and written comments from IT. We also received written comments from OPC, and Public and Indian Housing (PIH). These written comments are provided in Appendix A.

We held an extensive discussion on the draft report with IT on August 30, 1995. IT expressed concern that the language used and problems described in the draft report were exaggerated and inflammatory. Subsequent to the meeting, we considered IT concerns and made the changes that we deemed appropriate. IT comments, dated November 13, 1995, generally agreed with the recommendations in the revised version of the report but remained concerned that its comments and suggested language changes were

not fully addressed. We reviewed the comments and made further changes as needed. Reasons for not making certain suggested changes are discussed in Appendix A.

PIH suggested language changes to some of the recommendations. We have made the changes as deemed appropriate. OPC generally agreed with the intent of the recommendations.

Table of Contents

Abbreviations

ADP	Automated Data Processing
AQL	Acceptable Quality Level
ARN	Advance Requirements Notice
BPN	Budget Project Number
CA-JARS	CA-Job Accounting Resource Management System
CARMS	Computer Accounting Resource Management System
CASE	Computer Aided Software Engineering
CBSUP	Core Block Standard Unit of Processing
CHUMS	Computerized Homes Underwriting Management System (F17)
CLAIMS	Single Family Insurance System (A43C)
CMD	Computer Management Division
COBOL	Common Business Oriented Language
COM	Computer Output Microfiche
CPFF	Cost-Plus-Fixed-Fee
CPU	Computer Processing Unit
CSG	Computer Services Group
DB2	IBM Database Management System
DR	Difficulty Report
DSE	Dedicated Services and Equipment
FAR	Federal Acquisition Regulations
FAST	Funds Accounting and System Tracking
FIPS PUB	Federal Information Processing Standards Publication
FY	Fiscal Year
GAO	General Accounting Office
GSA	General Services Agency
HDS	Hitachi Data System
HIIPS	HUD Integrated Information Processing System
HQ	Headquarters
HUD	Housing and Urban Development
IAS	IT Inventory of Automated Systems

IEF	Information Engineering Facility
IPF	Interactive Processing Facility
IT	Office of Information Technology
IRM	Information Resource Management
IRMPB	Information Resource Management Planning Board
IRMWG	Information Resource Management Working Group
JCN	Job Control Number
LAN	Local Area Network
LOCCS	Line of Credit Control System (A67)
MAPPER	Application Development Tool for Unisys Mainframe
MFIS	Multifamily Insurance System (F47)
OFA	Office of Finance and Accounting
OFPP	Office of Federal Policy and Procurement
OIG	Office of Inspector General
OMB	Office of Management and Budget
OPC	Office of Procurement and Contracts
PAO	Program Area Office
PARMS	Project and Resource Management System
PAS	Program Accounting System (A96)
PC	Personal Computer
PCIE	President's Council on Intergrity and Efficiency
PCN	Project Control Number
PMS	Program Migration System
PRS	Performance Requirement Summary
PTARS	Problem Tracking and Reporting System
QAE	Quality Assurance Evaluator
QASP	Quality Assurance Surveillance Plan
QMS	Quality Management Staff
RMS	Resource Management Staff
SCM	Software Configuration Management
SDM	System Development Methodology
SEG	System Engineering Group
SEI	Software Engineering Institute
SHAS	Section 235 Subsidized Housing System (A65)
SMS	Software Migration System
SOW	Statement of Work
SVVP	Software Verification and Validation Plans
TAR	Teleprocessing Assistance Request
TIB	Technology Investment Board

TIIS	Title 1 Insurance and Claims
UAR	User Assistance Request
Unisys	Mainframe manufacturer
V&V	Verification and Validation
WCF	Working Capital Fund

Chapter 1

Introduction

According to Federal Information Processing Standards Publication (FIPS PUB) 106, software maintenance accounts for as much as 70 percent of the application software resources expended within the Federal Government. Current industry sources indicate that software maintenance can account for as much as 80 percent of an application system's lifecycle costs. Additionally, the rapidly growing inventory of software systems within the Federal Government is increasing the demand for software maintenance. It is therefore imperative that management has in place a strong, disciplined, and clearly defined approach to software maintenance, one that will assist management in controlling and improving the software maintenance process and provide accountability.

HUD has developed over 200 application systems to help manage varied and complex programs, which include: $378 billion of insurance in force and $12.9 billion in property and other assets related to the Federal Housing Administration (FHA) fund; $423 billion of Government National Mortgage Association (GNMA) Mortgage-backed securities; $25 billion in annual budget authority; and tens of billions of dollars of long-term housing subsidy commitments. HUD management classified loss of availability of 17 of these systems as a major risk for performance of the program mission, and another 25 as a serious risk.

Based on Fiscal Years 1993 and 1994 billings under the Working Capital Fund, HUD spent over $240 million in support of program activity controlling hundreds of billions of dollars in assets, insurance, and subsidies.

Audit Objectives

The audit objectives were to evaluate management controls over the application software maintenance process and the quality and quantity of cost information. Specifically, we evaluated HUD's:

- Software maintenance policies and procedures;
- Management of software maintenance during the system lifecycle;
- Management of contractors' performance of software maintenance; and
- Management of costs of application software maintenance.

Audit Scope and Methodology

Audit Scope

We judgementally selected seven operational mainframe application information systems for review. These seven systems were selected because: (i) they are considered critical or high risk sensitive information systems, as defined by the Office of Information Technology (IT), (ii) they are stable systems that have been in operational status ranging from 3 to 19 years, and (iii) the systems are representative of both the UNISYS and the IBM/Hitachi Mainframe platforms. Each system is briefing described below.

The A43C Single Family Insurance System-Claims Subsystem (Claims) is considered by HUD to be a major risk system with a criticality level of "needed immediately." This system supports the Department's Single Family Insurance Claim (SFIC) payment processes and provides online update and inquiry capability to SFIC data bases and to cumulative history files. Claim payments are made by check or Electronic Funds Transfer (EFT) daily via an Hitachi Data System/UNISYS/Treasury interface. For Fiscal Year 1994, this system processed 73,228 claims with a total dollar amount of $4.9 billion.

The A65 Subsidized Housing Accounting System (SHAS) is considered by HUD to be a serious risk system with a criticality level of "needed in the short term." The system is designed to provide accounting and financial management support for the Section 235 Mortgage Interest Assistance. For Fiscal Year 1994, this system processed an estimated 820 FHA cases with a total amount disbursed of $2.6 million.

The A67 Line of Credit Control System (LOCCS) is considered by HUD to be a major risk system with a criticality level of "needed immediately." The system is the Office of Finance and Accounting, General and Accounting Group's primary vehicle for cash management. It makes direct payments, either by check or wire transfer of funds, to recipients in response to submitted vouchers, yearly payment schedules, or telephone requests. For Fiscal Year 1994, 708,190 disbursements were made with a total dollar amount of over $26 billion.

The A96 Program Accounting System (PAS) is considered by HUD to be a serious risk system with a criticality level of "needed immediately." This system is an integrated budgetary accounting system for HUD's grant programs, including Community Development Block Grants, and is responsive to the financial needs of program accounting, budgeting, and auditing officials. For Fiscal Year 1994, this system processed at least $17 billion worth of obligations.

The F17 Computerized Homes Underwriting Management System (CHUMS) is considered by HUD to be a serious risk system with a criticality level of "needed immediately." This system assists and supports Field staff in the processing of single family mortgage insurance applications from initial receipt through endorsement. Additionally, the system provides assistance in appraisal and mortgage credit evaluation. For Fiscal Year 1994, a total of 1,338,296 endorsement cases were processed for a total mortgage insurance amount of $100.3 billion.

The F47 Multifamily Insurance System (MFIS) is considered by HUD to be a major risk system with a criticality level of "needed immediately." This system provides automated online, interactive support for HUD's multifamily mortgage insurance programs. It maintains the inventory of multifamily insurance-in-force cases, and all pertinent and historical data. During Fiscal Year 1994, this system processed 612 new insurance endorsement projects representing 101,896 units for a total amount of $3.2 billion.

The F72 Title I Insurance and Claims System (TIIS) is considered by HUD to be a serious risk system with a criticality level of "needed immediately." This system provides operational and management support for the execution of the Title I Property Improvement and Mobile Home Loan

Program. This includes loan inventory maintenance, billing, premium collection and reconcilement, claim processing, and Title I reserves maintenance and accounting. For Fiscal Year 1994, 72,148 in Title I loans were processed with a total dollar amount of $806 million and 7,602 Title I claims paid with a dollar amount of $65 million.

In addition to the seven systems identified above, we also reviewed 8 Automated Data Processing (ADP) software and maintenance contracts out of a total of 71 agencywide ADP contracts. These contracts provide software development and maintenance services for the Department, and were selected for their support of the seven systems we reviewed. The estimated contract value for these eight contacts is more than $135 million, of which approximately $8.6 million were billed for the seven applications during our review period.

Methodology

The audit steps included:

- Reviewing applicable Federal laws and regulations, industry practices, and HUD's policies and procedures, and practices related to our objectives;
- Interviewing HUD system managers and sponsors, program users, and IT, OPC, and contractor staff;
- Reviewing documentation related to project management and resource systems used for managing and controlling development and software maintenance efforts;
- Reviewing and analyzing data from the project management and resource tracking systems and comparing this data to the categories identified in FIPS PUB 106;
- Reviewing and analyzing reports from HUD's problem tracking and reporting system;
- Reviewing and analyzing HUD's software configuration management and change control process; and

- Reviewing and analyzing software development and maintenance contracts and related procurement documentation.

Audit Period

We performed the field work from March 1994 to November 1995. The field work reviewed cost, contract and performance data from the period October 1992 through March 1994. We conducted the audit in accordance with generally accepted government auditing standards.

Significant Control Weakness

The control deficiencies in software maintenance expose HUD to several high risks. Weak change controls could allow unapproved, unintentional, or malicious modifications to be introduced and proceed undetected through the change process, and software changes could be placed into production without adequate testing. Failure of any of the critical and high risk application systems we evaluated could prevent HUD from fulfilling its mission. As such, we consider these control weaknesses to be significant and merit prompt attention of agency senior management.

Chapter 2

Project Cost Accounting Must Be Used To Control Software Costs

HUD needs to implement a project cost accounting system that will enable HUD to collect and use cost information as intended by Office of Management and Budget Circular (OMB) Circular A-109. Our review of costs for the seven application systems disclosed the following:

- Approximately $4.7 million of the system engineering costs were categorized as development costs that would be categorized as either perfective or adaptive maintenance in accordance with FIPS PUB 106 guidelines;

- Approximately 66,800 system engineering hours, representing about $3.18 million, were not classified consistently between the project management and cost allocation and billing processes; and

- Software maintenance work associated with approximately $29.4 million of computer usage and telecommunications costs, were not separately identified.

We found that IT branch managers and program system sponsors were only managing approximately $11.2 million, or 27 percent, of the total $40.8 million costs billed for the seven systems reviewed. Also, HUD management had not periodically updated cost benefit analyses for major data systems as required.

These conditions exist because HUD lacks a project cost accounting system that can uniformly and consistently accumulate, track and report software costs for both management and billing purposes. Other reasons include an absence of a formal and consistently applied maintenance definition based on functional categories of maintenance activities as described in FIPS PUB 106. Segregating maintenance costs into these categories is essential for management to evaluate the costs and benefits

of system redesign versus continued maintenance of software which has become error prone, ineffective and costly. In addition, we found that HUD's policies and procedures do not require periodic cost benefit analyses as required by OMB Circular A-11 and A-130.

Consequently, HUD cannot make informed decisions regarding systems that cost more than $110 million a year to operate, develop, and maintain. Further, by not managing total lifecycle costs, HUD management can neither fully evaluate alternatives to satisfy system requirements nor determine when systems should be replaced because of excessive maintenance costs.

Federal Requirements and Guidance

OMB Circular A-109 requires that each agency acquiring major systems should maintain a capability to (i) predict, review, assess, negotiate, and monitor lifecycle cost; (ii) assess acquisition cost, schedule and performance experience against predictions, and provide such assessments for consideration by the agency head at key decision points; (iii) make new assessments where significant costs, schedule or performance variances occur; (iv) estimate lifecycle costs during system design concept evaluation and selection, full-scale development, facility conversion, and production, to ensure appropriate trade-offs among investment costs, ownership costs, schedules, and performance; and (v) use independent cost estimates, where feasible, for comparison purposes.

FIPS PUB 106, issued in 1984, is the most current Federal document that specifically addresses software maintenance. Although this document is over ten years old, the software maintenance concepts, functions, classifications, and processes identified in this publication still remain valid today. This document provides guidelines on controlling and improving software maintenance throughout a system's lifecycle. It also provides management with a definition of maintenance that functionally classifies software maintenance activities into three categories: perfective, adaptive, and corrective. Further, this publication concludes that improvements in the area of software maintenance will come primarily because of the software maintenance policies, standards, procedures and techniques instituted and enforced by management.

OMB Circular A-130 requires Federal agencies to account for the full costs of operating information technology facilities and recover these costs from the users. This Circular also requires Federal agencies to implement a system to distribute the full cost of providing services to the user. The term "full cost" is comprised of all direct, indirect, general and administrative costs incurred in the operation of the facility. These costs include personnel, equipment, software, supplies, contracted services, space occupancy, intra-agency and inter-agency services, and other services.

The following is a listing of the Federal and HUD requirements and guidance used besides the above to review the software maintenance cost portion of our audit.

- OMB Circular A-11, Preparation and Submission of Budget Estimates;
- 1994 GAO Executive Guide, entitled "Improving Mission Performance Through Strategic Information Management and Technology: Learning from Leading Organizations";
- HUD Handbook 2400.1, Information Resources Management (IRM) Policies; and
- *Applied Software Measurement, Assuring Productivity and Quality* by Capers Jones.

For more information about the criteria see Appendix B.

Resource Tracking and Billing Mechanisms

HUD uses the Working Capital Fund (WCF) to provide IRM services to all customers in the Department. Costs incurred by IT for staff and contractors are billed and paid by the customers through the WCF.

The Project and Resource Management System (PARMS) is the official project management system of the Software Engineering Group (SEG). Its purpose is to provide a formal means to record, monitor, and manage

all ADP systems work done by SEG staff or contractors. The entry of complete, accurate data about resources, projects, phases, activities, status comments, and time charges exists for two primary purposes: (1) to help managers oversee the process of building and maintaining automated systems, and (2) to generate time charge information that can be used to "charge back" system costs to users for budget purposes. The "charge back" system fulfills the OMB Circular A-130 requirement that the information processing facility recovers costs from the users of the facility. PARMS also has a budget function that management uses for budget formulation and tracking purposes, and to assist in the IRM planning and budgeting process. This budget module creates budget projects identified under individual budget project numbers (BPNs). BPNs contain general funding information, which mirror, at an aggregated higher level, the individual projects identified under project control numbers (PCNs). PCNs are used to record time charges. The BPNs and the PCNs are linked to the applicable WCF project number and reimbursable order number (RON) that are part of the Cost Allocation and Billing Process.

The Funds Accounting and Tracking System (FAST) is an internal system used by management for document preparation and tracking, and budget and contract accounting. Document preparation and tracking includes online entry of funding documents, online concurrence and approval of funding documents, and printing of electronically approved documents. Accounting functions include maintaining budget and contract balances as funding documents are approved, maintaining task specification balances as invoices are processed, producing financial reports for the Government Technical Representative, and tracking dollar expenditure by user for each subobject code.

Resource Tracking and Billing Mechanisms Are Overly Complex and Manually Intensive

A Project Cost Accounting System Is Needed to Manage Software Project Costs

HUD's process for tracking and billing costs is complex, disjointed, manual intensive, and does not interface with HUD's central accounting system. IT recovers WCF expenditures through reimbursements from customers' appropriated funds that flow through the Cost Allocation and Customer Billing Process (See Figure 1). The Resource Management Staff (RMS)

manages the cost allocation and customer billing process for the Working Capital Fund. This process combines project and budget information from PARMS with the contract and budget data from FAST, and resource usage data from the Computer Accounting Resource Management System (CARMS) and CA-Job Accounting Resource Management System (CA-JARS) to produce a monthly chargeback report that is submitted to the Office of Finance and Accounting (OFA). This report identifies charges by fund, customer, and RON. OFA uses the chargeback reports to transfer funds from program accounts to the Working Capital Fund to cover the costs and to record and update the General Ledger in the Agency Accounting System.

The process in Figure 1 also produces a monthly Statement of Services for each user's office. This statement is used to bill users for IRM services such as system engineering, mainframe computing, telecommunications, data entry and Computer Output Microfiche, and dedicated resources. Cost allocation allows IT to determine fair prices for the various services provided to its customers. Thus, the Customer Billing and Allocation Process serves to recover IT's costs for the IRM services provided.

The processing of cost allocation and customer billing data accumulated from PARMS and FAST is processed and billed using a series of Lotus spreadsheet files which require periodic manual updates to ensure the data is current and accurate. For example, when processing system engineering charges from PARMS, SEG prepares and submits to RMS, by cc:Mail, a PARMS extract ASCII file summary of Full Time Position (FTP) and contractor hours worked by customer, RON, WCF project number, and system. The data file is subsequently loaded in Lotus that updates an existing spreadsheet file. The hours are converted to dollar amounts using preestablished billing rates loaded in Lotus. This forms the basis for the system engineering amounts reported as part of the monthly chargeback report to OFA. OFA subsequently transfers funds from program accounts to the WCF to cover the costs and records and updates the General Ledger in the Agency Accounting System (AAS).

HUD is aware of the need to implement an automated project cost accounting and cost allocation system as part of the Agency Accounting System, but this effort was postponed. The Chief Financial Officer had planned to implement the Project Cost Accounting Subsystem

(PCAS)/Cost Allocation Application in FY 1995. However, this subsystem was deferred to FY 1996 because of a 35 percent cut in CFO's FY 1995 budget. We strongly urge that the project cost accounting subsystem be implemented as soon as possible. An automated project cost accounting system will provide information necessary for management to control software costs throughout its lifecycle.

HUD Needs To Develop A More Measurable Standard Software Chart of Accounts For Managing Project Costs

HUD does provide a software chart of accounts for both development and maintenance projects under PARMS. Development projects in PARMS are categorized into the six phases of the software development lifecycle, A through F, and identified as Initiation, Analysis, Design, Development, Testing, and Implementation, respectively. Each year, IT establishes a general maintenance project for each system for reporting most routine, low level maintenance efforts.

PARMS also permits project leaders to record three additional types of maintenance projects: Difficulty Report (DR), Technical Assistance (TA), and User Assistance Request (UAR). The "DR" maintenance type has not been used for several years. The "UAR" type is still in use but should be relabeled "PTAR" to describe the kinds of system maintenance problems identified by the Problem Tracking and Reporting System, A50, which replaced the UAR problem reporting.

However, the chart of accounts in PARMS does not provide the granularity needed for management to measure the economic impact of the major costs associated with software development and maintenance. The chart of accounts reflects phases of the software development lifecycle rather than specific activities (e.g. Requirements, Initial analysis and design, User documentation, etc.), that relate to a significant project milestone. Also, the chart of accounts does not accumulate costs by the FIPS PUB 106 maintenance categories (i.e. perfective, adaptive, and corrective). Consequently, management cannot adequately measure economic activity and productivity for software development and maintenance projects.

To more effectively manage IRM resources, HUD needs to establish specific measurable activities that provide management with an accurate

accounting of software development and maintenance projects. Chapter One of Capers Jones book on Applied Software Measurement identifies six key quantifiable data elements that affect software projects:

- The number of staff members assigned to a project;
- The effort spent by staff members;
- The schedule durations of significant project tasks;
- The overlap and concurrence of tasks performed in parallel;
- The project document, code, and test case volumes; and
- The number of bugs or defects found and reported.

Capers Jones indicated that, technically, one can adequately track and quantify the data by establishing a software chart of accounts for the tasks which will normally be performed on software projects and then collect data by task. However, he says the difficulty comes in recognizing that current tracking systems tend to omit large volumes of unpaid overtime, user effort on projects, managerial effort, and often many other activities.

Capers Jones states that one of the most frequent problems encountered with project historical data that lowers accuracy is a simple lack of granularity. Instead of measuring the effort of specific activities, the tendency is to accumulate only "bottom line" data for the whole project without separating the data into meaningful subsets, such as the effort for requirements, design, and coding. Additionally, Capers Jones states that, the breaking of software projects into specific phases such as "requirements, design, coding, testing, and installation," results in a cumbersome phase structure and is inadequate for cost measurement purposes. Too many activities, such as production of user documentation, tend to span several phases, so accurate cost accumulation is difficult to perform.

Table I provides an example of the standard chart of accounts (i.e. cost accumulators) illustrated in Capers Jones book when collecting project data. It illustrates the kind of granularity by activity, rather than phases,

that is needed for historical data to be useful for economic studies. An "activity" is defined as a bounded set of tasks aimed at completing a significant project milestone (e.g. completion of requirements, completion of a prototype, completion of initial design, etc). Although Table I uses standard activities, it does not imply that only 25 things must be done to develop a software package. Any given activity, such as requirements, consists of various significant subactivities or specific tasks. However, Capers Jones contends that a task-level chart of accounts tends to be very cumbersome for cost accumulation since project staff must record times against a very large number of individual tasks.

Cost Accounting Standards Need To Be Established For Software Development and Maintenance Projects

Cost accounting standards for software development and maintenance projects must be developed. Cost accounting standards provide uniformity and consistency in the estimating, accumulating, allocating, and reporting of project costs. Uniform and consistent application of cost accounting standards throughout the project tracking and customer billing and allocation processes provides management a means to qualitatively and quantitatively evaluate and measure results against established goals.

We noted, however, that HUD does not have cost accounting standards for software development and maintenance activities. The Director, Office of Financial Policy and Evaluation stated that the implementation of cost accounting standards, although needed in the long term, is not a high priority item. However, cost accounting standards must be adopted so IT system managers and program system owners can manage their IRM costs more efficiently and effectively.

Maintenance Costs Are Not Properly Categorized and Consistently Applied

$4.7 of System Engineering Software Maintenance Costs Are Not Categorized In Accordance With FIPS PUB 106

FIPS PUB 106 indicates that software maintenance accounts for approximately 60 to 70 percent of the application software resources expended within the Federal Government. FIPS PUB 106 also categorizes software maintenance into three functional classifications:

perfective, adaptive, and corrective.

As part of our review, we made a determination to identify the maintenance costs for the seven systems based on the software maintenance definition and functional classifications of FIPS PUB 106. We believe that these functional classifications are essential for HUD to properly manage costs related to software maintenance (See Chapter 3). Project charges from PARMS are reported to RMS as either "development" or "maintenance" categories. Accordingly, we asked the various IT branch managers to review the project charges reported in PARMS for their respective systems over the 18-month period ending March 31, 1994, and to reclassify the charges based on the definition and functional classifications as identified in FIPS PUB 106. The results of this review disclosed that the maintenance costs in PARMS differed by $4.7 million for the seven systems (See Figures 2 and 3).

NOTE: Figure 2 shows the comparison of maintenance costs in system engineering that were recorded in PARMS against what the costs should have been using the functional classification criteria of FIPS PUB 106. Results show a $4.7 million difference in maintenance cost amounts. The dollar amounts were derived by taking the hours in PARMS for the seven systems and using a fully loaded standard rate of $47.50 and $48.00 that was used for billing under the WCF for FY 93 and FY 94, respectively.

NOTE: Figure 3 shows maintenance costs compared to development costs based on PARMS versus FIPS PUB 106 classifications. The chart shows that maintenance costs per FIPS PUB 106 functional classifications are almost the inverse of the costs recorded in PARMS and are more in line with the 60-70 percent Federal agencywide standard as indicated in FIPS PUB 106.

The primary reason for the significant difference in maintenance costs is that projects are categorized as either maintenance or development based on criteria and classifications that are not functionally segregated and consistent with FIPS PUB 106. Further, we noted that the classifications used in both the project tracking and customer billing and allocation processes were in themselves inconsistent.

Under PARMS, "maintenance" projects are classified as those efforts to keep the system running, to correct latent defects or to assist the user (i.e. corrective maintenance per FIPS PUB 106). "Development" projects are classified as work to create a new system or to modify an existing system to provide new functions (i.e. perfective and adaptive maintenance per FIPS PUB 106). At the beginning of each fiscal year, a PARMS auto-generated Project Control Number

(PCN) is established to create a general maintenance project for each system in production. Project leaders are given broad latitude to use this project to record system maintenance work performed on each system. In addition, project leaders are also sometimes asked to establish a separate maintenance project PCN to record work performed to resolve system Difficulty Reports (DR) or User Assistance Requests (UAR) or to provide Technical Assistance (TA). All other system engineering work is performed under separate "development" PCNs. Development projects are projects initiated by an Advanced Requirements Notice (ARN) or those classified as Non-ARN. ARNs are submitted by the user requesting development of a new system or the enhancement (i.e., perfective maintenance) of an existing system.

Further compounding this issue is the effect the IRM budget process has on the categorization and classification of maintenance versus development costs. Various IPS system managers and staff informed us that how projects are recorded as maintenance or development is driven by IRM budget and funding categories rather than any functional classifications. The maintenance and development budgets were set by the Information Resource Management Planning Board/Technology Investment Board (IRMPB/TIB) (Auditors Note: The IRMPB became the TIB in FY 1994 with the functions and responsibilities remaining the same). Maintenance budget projects are established and funded based on an approved "Business Maintenance" discretionary funded line item in the IRM budget. However, "Business Maintenance" at the IRMPB/TIB level includes "fix it if it's broken" repairs and institutional changes (i.e. corrective and adaptive maintenance). All other IRMPB/TIB approved priority projects, which include system enhancements and modifications (i.e. perfective and adaptive maintenance), are considered development type work and therefore categorized under "development" type budget project numbers.

Consequently, most of the "maintenance" projects classified in PARMS and in the IRM budget process are considered corrective maintenance per FIPS PUB 106, whereas any enhancements and modifications (i.e., perfective and adaptive maintenance per FIPS PUB 106) are classified as "development" projects. This results in a significant difference in the maintenance costs associated with each system as reported under PARMS from what the maintenance costs would be had HUD used the FIPS PUB 106 categorizations.

In times of budget cuts, managers are required to make painful decisions about deferring software maintenance. Corrective and adaptive changes frequently

cannot be deferred. This leaves perfective changes as the source of discretionary changes, which may have to be deferred for budgetary reasons. If managers cannot identify which changes are corrective, adaptive, or perfective, they cannot make effective decisions about software maintenance funding priorities.

$3.18 Million in Maintenance and Development System Engineering Costs Were Not Consistently Classified

Our review disclosed inconsistencies between what was classified as development and maintenance system engineering costs under the project management (PARMS) process and what was classified and reported to HUD users under the WCF Statement of Services as part of the Cost Allocation and Customer Billing process.

RMS receives a monthly PARMS report that identifies direct FTP and contractor labor hours by user, RON, WCF project number, system, contract and task. The report further breaks down the direct labor hours by "development" and "maintenance" categories. These labor hours are converted to costs based on a predetermined fully loaded hourly billing rate. IT uses this data to bill the users shown in the WCF Statement of Services and to prepare the chargeback report for OFA.

As part of our review, we reconciled and analyzed PARMS data, FAST data, and WCF Statements' of Services for the 18-month period ending March 31, 1994 for the seven application systems. We traced and reconciled individual projects under each of the seven systems to the applicable BPN and to the appropriate project number and RON under the WCF.

As illustrated in Table II, we found a total of 66,843.4 hours identified under development and maintenance budget categories in PARMS that were not consistently classified with the RONs and project names under the WCF. Consequently, $3.18 million of maintenance and development costs were inconsistently classified between the project management and the WCF billing and reporting processes.

Our analysis disclosed that 5,559.5 hours, with a dollar equivalent value of $266,856, were reported as "Development" type system engineering project work under six BPN's in PARMS. However, of these six BPNs:

- Three (94-BM01-A43C, 94-BM02-AF97, and 94-BM02-CC97) were classified and billed under the "Business Maintenance" category both in PARMS and under the WCF;

- Two (94-BD02-F47 and 94-BD03-F47) were classified under a "development" budget category (i.e. BD) in PARMS, but were classified, reported and billed in the WCF as "Business Maintenance;" and

- One BPN (94-BM01-HS97) was classified in PARMS as "Business Maintenance" but under the WCF, it was classified, reported, and billed as "System Development."

Additionally, our analysis also showed that 61,283.9 hours, with a dollar equivalent value of $2,913,349, were reported as "maintenance" type work under 10 BPNs in PARMS but were classified, reported, and billed under the WCF as "System Development" work.

Maintenance Costs for Computer Usage and Telecommunications Must Be Tracked

For the 18-month period ending March 31, 1994, computer usage and telecommunications costs accounted for $29.4 million, or 72 percent, of the total $40.8 million billed to HUD customers for IRM services for the seven systems reviewed. We found, however, the current charge back system does not separately identify computer usage and telecommunications costs associated with software maintenance. As a result, HUD management does not have the total software maintenance costs needed to make informed decisions on whether to continue maintaining certain systems.

Monthly reimbursable costs for centralized information processing services are billed to IT customers through the WCF Statement of Services. Computer usage and telecommunications are reported under the Mainframe Computing and Telecommunications resource categories in the customer Statement of Services. Core Block Standard Unit of Processing (CBSUP) hours are used as the service units under the Mainframe Computing and Telecommunications categories for the Unisys mainframe platform. Computer Processing Unit (CPU) hours and costs are used as the service units under the Mainframe Computing and Telecommunications categories, respectively, for the Hitachi

mainframe platform.

CARMS is used to track and report computer usage charges on the Unisys mainframe platform. CA-JARS is used to track and report computer usage charges on the Hitachi mainframe platform. Under CARMS, computer usage is segregated under only two computer accounts of production and development. Computer usage from CA-JARS is reported based on various descriptive accounts that are segregated primarily under the categories of development, production, and testing. CBSUP hours are used for identifying combined CPU time and file storage based on CARMS data. Data obtained from CA-JARS identifies separate CPU time (hours) and file storage (Megabytes) for reporting computer usage. However, neither CA-JARS nor CARMS separately identify and report computer usage related to software maintenance.

IT is currently in process of implementing a BILLER-1100 computer resource application package. This package is intended to replace CARMS and interface with CA-JARS to provide a baseline off-the-shelf resource accounting package which will form the hub of an integrated data collection system for computer resource usage. In conjunction with this implementation, IT needs to establish a maintenance account in this system which will be used to segregate and report software maintenance computer usage. This is necessary since computer usage and telecommunication costs represent a significant portion of HUD's software engineering efforts. Without identifying and segregating maintenance costs from development costs for computer usage and telecommunications, management will not have the complete cost information to make informed management decisions on whether to continue maintaining systems or whether certain systems should be considered for redesign.

Lifecycle Costs Are Not Being Managed

Only System Engineering Work At The Application Level Is Being Managed

OMB Circular A-109 requires that each agency maintain the capabilities to predict, review, assess, negotiate, and monitor lifecycle costs. Lifecycle costs are defined as the total of the direct, indirect, recurring, one time, and other related costs, incurred or estimated to be incurred, in the design, development, production, operation, maintenance, and support of a major system over its anticipated useful life span.

HUD Handbook 2400.1, Information Resources Management Policies (IRM) provides that IT is responsible for the direction and execution of HUD's IRM program which includes, in part, reports management, computer operations and maintenance, ADP budgeting and resource management, systems engineering services, and long-range IRM planning. The Primary Organization Heads (i.e. system users) and their staffs forecast future automation needs and priorities and play key roles in the ADP planning and budgeting process in support of those needs. The system users assess and identify their ADP requirements and provide initial specifications for ADP systems and work in close coordination with IT during the development, implementation, and ongoing operational activities of these systems. The system sponsor is the HUD organizational element where the functions of management and control of a particular system reside. The sponsoring organization is responsible for system planning and budgetary support. The IT branch manager is responsible for assisting users with defining and providing cost estimates of future system requirements.

For the 18-month period ending March 31, 1994, approximately $40.8 million was billed to HUD system users under the seven major information systems we reviewed. However, we found that neither the IT branch managers nor system sponsors could provide us with the total lifecycle costs for their respective systems. Additionally, we noted that a only small portion of the $40.8 million of costs billed to the users of the seven systems were being managed.

Our review disclosed that only system engineering work at the application level is being managed. This work accounts for about $11.2 million, or 27 percent of the total $40.8 million of billed costs. Computer usage, telecommunications, and other costs, which make up the remaining 73 percent of the total costs, were not being managed by either the IT branch managers or the system sponsors (See Figures 4 and 5).

We were told by the system sponsors that they rely on IT to manage system lifecyle costs and to provide them with cost estimates for system work performed. One system sponsor, responsible for four of the seven systems that we reviewed, indicated that they rely on IT to manage system engineering, CPU time and other technical costs. Also, another system sponsor, responsible for three of the systems we reviewed, indicated that they do not collect lifecycle costs at their level and rely on IT to provide them with cost estimates for any system development or major enhancements. When we discussed this issue with the IT branch managers, however, we were told that the IT branch

managers do not manage total lifecycle costs. Further, the IT branch managers indicated they only manage contractor and in-house system engineering costs and do not manage the computer usage, telecommunications, and other costs. By not adequately managing all system lifecycle costs, system managers cannot properly plan and budget their IRM requirements or manage their projects in the most cost effective and efficient manner.

NOTE: Figure 4 shows for each application the four cost categories billed to the WCF and the associated costs. The IT branch managers and program system sponsors only track system engineering costs, which represent only a portion of the total costs.

NOTE: Figure 5 shows the totals of the four cost categories from Figure 4 in a pie chart. The system engineering costs managed by IT branch managers are cut out from the pie. The remaining 73 percent of the costs are broken down on the right of the chart. This chart shows that three of the four cost categories are not being managed by either IT branch managers or the system sponsors.

Cost Benefit Analyses Not Periodically Updated

OMB Circular A-11 requires agencies to prepare cost benefit analysis following OMB Circular A-94 for all proposed investments, including those for application systems. OMB Circular A-109 requires that periodic assessments be made where significant costs and performance variances occur and provide such assessments for consideration by the agency head at key decision points in the lifecycle process. Also, OMB Circular A-130 requires agencies to prepare cost benefit analyses for all application systems at a level of detail appropriate to the size of the investment and update as necessary throughout the application system's lifecycle. Further, HUD Handbook, 2400.1 REV 1, Information Resources Management (IRM) Policies provide that cost benefit analyses, including analyses of several alternatives, be performed for new information initiatives and existing system modifications that are projected to cost more than $5 million totally or more than $2 million in a single year.

Although HUD regulations require cost benefit analyses be performed, we found that cost benefit analyses for major ADP projects are not periodically updated as required by OMB Circular. The purpose of the cost benefit analysis is to ensure that the most cost effective alternative that satisfies system requirements is chosen. Without this analysis, it is not possible for system managers to evaluate alternatives to satisfy their system requirements.

We found that none of the IT branch managers and system owners we interviewed could provide us with a current, updated cost benefit analysis for their respective systems, though three of the seven systems, A43C, A67 and F17

incurred costs exceeding $5 million during the 18-month review period. Various IT system managers and system sponsers told us that a cost benefit analysis is only performed when a system is initially developed and when major enhancements or modifications are planned.

Although HUD's current policies and procedures require cost benefit analyses to be performed, the policies and procedures do not address the need to keep these analyses current. A majority of the IT branch managers explained that the reason cost benefit analyses are not updated was because most of the major enhancements and modifications are made due to changes in legislation and, therefore, would have to be done regardless of the cost involved.

The reason cited for not updating cost benefit analyses is not valid. While we realize that legislative requirements must be implemented, this does not preclude one from considering the costs and economic and intangible benefits of several alternatives for satisfying a particular requirement. For example, when new requirements are placed on a system, besides modifying the existing system, other alternatives such as building a new system or buying a commercial software package should be considered. It may be more efficient and economical to buy or use already developed software than continually modifying an existing application to satisfy changing requirements. These cost benefit evaluations, with other studies, provide managers and users with the information necessary to make the most efficient and cost effective decisions on the allocation and use of limited IRM funds.

Recommendations

Chief Financial Officer

2 (a) Expedite the implementation of the Project Cost Accounting System (PCAS)/Cost Allocation Application.

2 (b) Establish cost accounting standards to ensure maintenance and development project costs are uniformly and consistently accumulated, tracked, allocated, and reported for management and billing purposes.

Acting Director of Information Technology, Through the Technology Investment Board

2 (c) Establish a software maintenance definition that can be uniformly and consistently applied throughout HUD's project accounting and cost allocation and billing processes that meet the intent of FIPS PUB 106;

2 (d) Establish appropriate software maintenance classifications that, as a minimum, include the three categories described in FIPS PUB 106;

2 (e) Establish a more measurable software chart of accounts that describe specific development and maintenance activities needed to gain economic insights into the major costs associated with software; and

2 (f) Establish guidelines and define management responsibilities for capturing all system lifecycle costs to include updating the required cost benefit analyses at appropriate, defined, and periodic intervals.

Acting Director of Office of Information Technology

2 (g) Establish a descriptive maintenance account within the integrated resource accounting system Biller 1100 to capture software maintenance computer usage costs.

Auditee Comments and OIG Response

In its response to our draft report, IT fundamentally disagreed with OIG's conclusions on using FIPS PUB 106 classifications (i.e. adaptive, perfective, and corrective) when tracking maintenance costs and activities throughout the systems lifecycle. During the August 30, 1995 meeting with IT, the OIG agreed with IT that the terminology used for classifying maintenance costs could be different but the underlying FIPS PUB 106 classifications and definitions are still valid. This is supported by the fact that organizations, who are serious about cost containment and measuring software processes and their business value, utilize these same FIPS PUB 106 classifications. For example, the September 1995 issue of IT Metrics Strategies, identifies perfective, adaptive, corrective, and preventive maintenance work types for use in measuring the amount of application development and maintenance work performed by an organization. Also, as noted in Appendix E, a recent study that analyzed the software maintenance process in the European space industry used the same classifications but in a more precise and specialized way. We do not suggest

that these classifications are all inclusive. Rather, IT should use these classifications as a starting point and develop, if necessary, additional work type categories and classifications for managing their software lifecycle costs.

Chapter 3

Numerous Controls Needed For Application Software Changes

HUD spends over $110 million each year to provide information systems services, including operation, development, and maintenance of application systems supporting program activity controlling hundreds of billions of dollars in assets, insurance, and subsidies. Controlling the changes to the software of these systems is essential to keep them functioning and responsive to user needs. OMB Circular A-130 makes management of information systems a shared responsibility. The program manager is responsible for obtaining the information system(s) necessary to fulfill the mission, and the information technology service provider must provide adequate technical support. Despite Federal guidance, we found management weaknesses in all seven HUD application systems we reviewed. Program and functional offices, as system owners, have not: (1) established centralized approval and review of changes, (2) developed a maintenance schedule (3) defined standard classification of changes, and (4) performed sufficient user testing and acceptance of software changes. In addition, the Office of Information Technology (IT) has not defined the software maintenance process, measured the performance of maintenance activities, or defined quality assurance goals. IT also has not implemented a process for systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration throughout the software lifecycle.

Deficiencies in software change controls have resulted in a significant amount of production problems during maintenance. An IT study of change releases concludes that release-related problems are costly and that release-related activity represents an important cost factor to some systems, including two of the systems in our audit. The study suggests that costs related to release activity and release-related problems be reported to the users. The results of the IT study suggest that established review and testing procedures fail to prevent release-related problems, and perhaps contractor performance could be improved (see Chapter 4). These errors could also indicate that these systems have become unstable or unreliable and need to be replaced.

The deficiencies in managing software changes are attributable to system owners not taking responsibility for managing application systems needed for successful conduct of the program mission. These deficiencies are exacerbated by a lack of a software maintenance policy. Additionally, performance measures have not been established to aid management in determining the quality of software after changes are made.

Focus of the Change Control Finding

We collected and analyzed release data for seven application systems. While there have been instances of problems attributable to deficient management of software changes, we agreed with the IT managers that analyzing the performance data in aggregate, to determine what preventive controls are needed, is more beneficial than focusing on isolated past incidents.

We have also used tables and charts in this Finding to illustrate possible measurements that can be used for evaluating software maintenance. These measurements include the number of release requests for the same change request, number of changes to a program, number of problems reported, etc. We offer these examples as a starting point for defining performance measures. Appendix C discusses in detail the software measurement process, and presents information on how the process works and how it can be used by HUD managers to control quality and productivity.

OMB Circular A-130 states that program managers are responsible for acquiring the necessary technical and operational support of their application systems to carry out their programs. A technical support organization such as IT has a responsibility to meet their service and infrastructure commitments to their program clients. This makes a process such as software maintenance a shared responsibility, with IT providing the infrastructure and the software maintenance service, and the program offices responsible for managing the product--the information system being maintained.

Federal Requirements and Guidance and Other Criteria

We used the following list of Federal requirements and guidance to conduct our audit in the area of software change control and configuration management. We also used a number of publications on industry-accepted practices to evaluate software maintenance activities. Appendix B contains a summary of the following criteria:

- Public Law No. 99-511 (94 stat 2812), *The Paperwork Reduction Act of 1980*;
- Public Law No. 99-591 (100 stat 3341-335), *The Paperwork Reduction Reauthorization Act of 1986*;
- Public Law No. 103-62 (107 stat 285), *The Government Performance and Results Act of 1993*;
- OMB Circular No. A-130, *Management of Federal Information Resources*;
- FIPS PUB No. 38, *Guideline for Documentation of Computer Programs and Automated Data Systems*;
- FIPS PUB No. 106, *Guideline on Software Maintenance*;
- FIPS PUB No. 132, *Guideline for Software Verification and Validation Plans*;
- National Bureau of Standards Special Publication 500-129, *Software Maintenance Management*;
- *GSA Guide for Acquiring Software Development Services*;
- HUD Handbook 1100 series, *Administration*;
- HUD Handbook 2400 series, *Information Resources Management*;
- *Applied Software Measurement, Assuring Productivity and Quality* by Capers Jones;
- *Making Software Measurement Work, Building an Effective Measurement Program* by Bill Hetzel; and
- Capability Maturity Model for Software, version 1.1, Software Engineering Institute.

System Owners Must Manage Their Software Changes

OMB Circular A-130 states that program managers are responsible for acquiring the necessary technical and operational support of their application systems to carry out their programs. Our review of software maintenance finds that program offices rely on IT for software maintenance services and exercise only limited management over the service. Program offices do not routinely monitor the changes that they are requesting to an information system or classify changes to facilitate historical reviews. Maintenance activities are not scheduled to provide stability and predictability. Finally, program offices do not routinely evaluate test results.

Process is how we go from the beginning to the end of the project. Key processes associated with the management of software development and maintenance include requirement management and software configuration management. Under requirements management for each information system, responsibility is established for analyzing the system requirements and allocating them to hardware, software and other system requirements.

Software Configuration Management (SCM) implies that a board having the authority for managing the project's software baseline (i.e., a software configuration control board--SCCB) exists or is established (IT should participate on the board as a technology consultant). Members of the SCM group are trained in the objectives, procedures, and methods for performing their SCM activities. The software work products to be placed under configuration management are identified. Changes to baseline are controlled according to a documented procedure. Change requests and problem reports for all configuration items are initiated, recorded, reviewed, approved, and tracked according to a documented procedure.

Centralized Change Review Is An Important Control

According to FIPS PUB 106 a key to controlling changes to an application system is the centralization of change approval and the formal requesting of changes. A centralized approval process within the application owner organization will enable one person or a group of persons to gain the knowledge of all the requested and actual work being done on the system.

None of the seven application systems we reviewed had a centralized change review process. A centralized review process can prevent the following from occurring:

- Similar enhancements to the system are processed individually rather than combined, which results in wasted resources from duplication;
- A proposed software change is made without evidence of an audit trail;
- A proposed software change is made without sufficient impact analysis and creates problems for other software components of the application;
- A software change is made that provides limited benefits; and/or
- A proposed change of lesser importance is made ahead of more significant and necessary modifications.

Historical Review Can Help to Identify Problems

One of the important activities necessary to control change is the analysis of problem reports and software changes. We found no instances where system owners periodically conducted systematic reviews of software changes to discern performance patterns or trends. Available tracking systems did not contain a level of data sufficient to facilitate informative trend analysis of the effects of prior software changes.

The analysis of problem reports and software change requests provides an informed perspective on the performance trends of the information system. A formal process is needed to ensure that requests for system changes, as well as system problems, are reported and documented in a standard manner. The data gathered by a change request and problem reporting system can be used to describe the effects of software changes on the quality of a system. The benefits of using historical change control and problem data enable managers to: (1) identify modules that experience a high degree of problems requiring corrective changes; (2) identify modules repeatedly reworked due to deficiencies in contractor performance; (3) identify modules requiring rework

due to problems in testing and review procedures; or (4) determine whether it would be more cost effective to redesign than continue maintaining these modules.

IT Study Can Be Used As a Model For Historical Review

The IT Quality Management Staff (QMS) provides a model of the type of analysis program officials can make for each application system supporting the program mission. From September 1994 through December 1994, QMS conducted a review of the processes and procedures used to install software changes or releases[4] on the HUD mainframe computers. The major objectives of the study were to understand how well the Standard Release Procedures are working, to identify and understand relationships between software releases and subsequent production problems, and to recommend actions that might improve processing of software releases. This review included:

- Interviews with IT and contractor staff;
- Examination of all release requests submitted to Computer Management Division (CMD) between January 1 and August 31, 1994; and
- Review of Problem Tracking and Reporting System (PTARS) and release requests associated with release-related problems.

The review disclosed the following:

- During this period there were 158 routine releases (42 percent) and 218 emergency releases[5] (58 percent);
- 51 percent of all release requests indicated that the release was intended to correct a problem;
- 16 percent of all releases resulted in a problem;
- 63 percent of the problems resulted from emergency releases;
- 42 percent of the problem releases were from releases submitted within the same day or within 24 hours of the requested release date; and

- Of the releases studied, 26 caused a total of 44 additional releases.

The review reached the following conclusions:

- Majority of the software releases did not result in production problems;
- Release-related problems are costly, and should be reduced;
- Release-related activity represents an important cost factor for some application systems; and
- Additional attention is needed to improve management of releases.

While the IT review covers a shorter time span within the scope of our review, they have taken a broader sample of applications and completed a more thorough review than we have. This was a comprehensive review which achieved its objectives and produced valuable insights into software release procedures. Therefore, we have relied on IT's detailed analysis of these release requests as well as our own analysis to provide indicators of potential problems. However, the IT objective of recommending actions to improve processing of software releases limits the benefits of the analysis. Using the study to focus on improving the software change process would automatically reduce release-related production problems.

Another Example of Historical Review

Experience has shown that sources of potential efficiency improvements and sources of errors are rarely uniformly distributed across all modules of a system. Routine assessment of historical information would help system owners to identify programs or processes that may be candidates for redesign. For each information system we reviewed, we analyzed the release elements of the CMD Release Request, and tallied the number of times each program was placed into production. We have used the Computerized Homes Underwriting Management System (CHUMS) to illustrate a system in which several key programs are affected by every system change, and not to suggest that we have identified a problem with CHUMS programs.

NOTE: Figure 1 illustrates the software modules most frequently changed for the F17 CHUMS application. During this period, 275 changed modules were placed into production. The 11 CHUMS modules shown in Figure 1 account for 80, or 30 percent, of the module changes.

Without routine assessment of historical information, system owners cannot:

- Determine existing maintenance problems or detect their probable cause(s);
- Assess the overall stability of the system;
- Identify programs or modules that are routinely modified/reworked;
- Make informed decisions about the future of the system, its impending obsolescence, or necessary redesign of the application system; and
- Assess whether unnecessary or inefficient effort was expended or rework is evident for previously reported problems. Both could contribute to excessive maintenance costs.

Lack Of Scheduled Maintenance

IT has standard procedures for releasing application software to the HUD Integrated Information Processing System (HIIPS) production environment. The schedule of releases is on a Tuesday-to-Tuesday cycle. To release software changes on Tuesday, the request must be received by the previous Tuesday. Any request outside of this cycle constitutes an emergency. However, this schedule provides stability for the HIIPS production environment but not the application systems of the individual program office. All seven applications we reviewed lacked planned maintenance where software changes are placed into production following a schedule.

Scheduled maintenance activities furnish users with periods of stable operation and expected system performance. When changes are planned and implemented according to a maintenance schedule, users can be informed of pending changes ahead of time and receive appropriate instruction on new or revised operating procedures or functional capabilities. Additionally, limiting implementation of

software changes to regularly scheduled events enables management to maintain version control over the numerous modules/programs that comprise each of the application systems.

Without a schedule, maintenance can result in continuous software modifications which contribute to the instability of an application system. When a system of interrelated programs, modules, and tables are continuously modified, ensuring proper and effective version control becomes difficult. This can complicate a return to a prior software version should a software change fail after being placed into production.

NOTE: Figure 2 shows the software changes to F17 CHUMS from 1/1/93 to 6/30/93. The project leader indicated that the spikes on the chart correspond to new releases. Frequently the temporary increase in PTARS represents field office users unfamiliarity with the revised system.

Figure 2 provides an example of a historical review which shows the need for scheduled maintenance. For each information system we reviewed, we analyzed the release elements of the CMD Release Request, and the PTARS associated with the system. Graphically presenting software releases against a background of PTARS may provide indications of the impact of new operating procedures, changing data input requirements, varying screen presentations, and new or modified system functions or reporting capabilities. Figure 3 is an example of random releases without a maintenance schedule which can cause operational problems for the users.

NOTE: Figure 3 shows the release of F72 TIIS software into production and the number of release elements associated with each release. Note the near continuous release of small changes to the software.

No Standard Classification Of Changes

System owners have not classified changes in a manner that would facilitate analysis of software maintenance trends. None of the seven application systems we reviewed had used the classification types suggested by Federal criteria for categorizing the change requests. Instead, the change requests were based on the level of effort involved with the maintenance activity and arbitrarily assigned either as development or maintenance projects (see Table 1 of Chapter 2).

We reviewed CMD Release Requests for the seven systems in our review. We looked for indications that the release was justified as a correction to a prior release (omission, restoration of old version, release). A summary of this analysis is presented in Table I. Table

I identifies some of the effects that were discussed in the previous paragraph. While this is not a comprehensive or statistically valid sample, the analysis shows enough rework to justify closer examination. FIPS PUB 106 indicates that approximately 20 percent of all changes are error correction, 20 percent are initiated in response to changes in the environment (e.g., legislative changes), and the remaining 60 percent are made to meet the evolving needs of the users.

NOTE: The correction releases in Table I refer to emergency corrections to software that have just been tested and approved for release into production operation, but failed to work properly. We have made no attempt to identify all correction releases, but only those identified by the project leader as corrections on the CMD Release Request form.

Insufficient Testing And Acceptance By Functional Users

Out of seven IT project leaders we interviewed, only two obtained review and approval of test results from program office counterparts before requesting the changes be made. Normally the IT project leader reviews and approves the test results. Responsible program offices in general do not perform a functional assessment of the modified software to ensure: (1) only authorized work was completed; and (2) that all requirements of the change request had been met and the system functioned according to specifications. Without a functional acceptance test, program managers have no assurance that perfective or corrective changes made will satisfy the user needs when placed into production. This increases the risk that the changes will waste time and resources.

Commonly, functional users do "acceptance[6]" testing after system testing has been completed. The software maintenance process is not considered complete until the user has accepted the modified system and all documentation has been satisfactorily updated.

IT Must Manage Software Maintenance Process

IT provides information management services in Headquarters, Regional, and Field Offices. The office provides ADP systems development and maintenance services, including vendor-developed software packages. Our review finds that software maintenance services are delivered in an informal and inconsistent manner. IT has not defined software maintenance consistently with Federal guidelines; adopted a single change request form; established a tracking process based on measurement; conducted adequate testing of software changes; performed independent verification and validation testing; or established formal software configuration management even though IT has purchased the

ENDEVOR configuration management tool.

OMB Circular A-130 states that a technical support organization such as IT has a responsibility to meet their service commitments to their program clients. FIPS PUB 106 states that management is clearly one of the most important factors in improving the software maintenance process. A complete management system includes a performance monitoring feedback loop with successive cycles of goal setting, performance monitoring based on measurement, analysis of results, review against the goals, and reporting.

FIPS PUB 106 states management must examine how the software is maintained, exercise control over the process, and ensure that effective software maintenance techniques and tools are employed. The Software Engineering Institute defines key process areas whose implementation leads to quality improvements in the process of delivering information systems. Key processes corresponding to FIPS Pub 106 requirements for management of software maintenance are software tracking and oversight, software quality assurance, and software configuration management.
Software tracking and oversight involves tracking and reviewing the software accomplishments and results against documented estimates, commitments, and plans, and adjusting these plans based on the actual accomplishments and results. A software maintenance plan is prepared, managed, and controlled (i.e., version control and change control on the plan) for tracking the software activities and communicating status. Approved changes to commitments that affect the software project are communicated to the members of the software engineering group and other software-related groups.

Project tracking implies measurement and the tracking of costs at a detail level. The Software Engineering Institute suggests several relevant measurements, for example:

- The size of the software work products (or the size of the changes to the software work products) are tracked, and corrective actions taken as necessary;

- The project's software effort and costs are tracked, and corrective actions taken as necessary;

- The project's critical computer resources are tracked, and corrective

actions taken as necessary;

- The project's software schedule is tracked, and corrective actions taken as necessary;

- Software engineering technical activities are tracked, and corrective actions are taken as necessary;

- The software risks associated with cost, resource, schedule, and technical aspects of the project are tracked; and

- Actual measurement data and replanning data for the software project are recorded.

The software engineering group should conduct periodic internal reviews to track technical progress, plans, performance, and issues against the software development plan. Formal reviews to address the accomplishments and results of the software project are conducted at selected project milestones according to a documented procedure. Software project commitments and changes to commitments made to individuals and groups external to the organization are reviewed with senior management according to a documented procedure.

Software quality assurance provides management with visibility into the process being used by the software project, and the products being built by the project. Software quality assurance activities assure that adherence of software products and activities to the applicable standards, procedures, and requirements is verified objectively. A Software Quality Assurance plan is prepared, managed, and controlled (i.e., version control and change control on the plan) for each information system according to a documented procedure. Affected groups and individuals are informed of software quality assurance activities and results. Noncompliance issues that cannot be resolved within the software project are addressed by senior management.

Software configuration management involves identifying the configuration of the software; systematically controlling changes to the configuration; and maintaining the integrity and traceability of the configuration throughout the software lifecycle. A Software Configuration Management plan is prepared, managed, and controlled (i.e., version control and change control on the plan) for each information system according to a documented procedure. A

configuration management library system is established as a repository for the software baseline. Products from the software baseline library are created, and their release is controlled according to a documented procedure. The status of configuration items is recorded according to a documented procedure. Standard reports documenting the SCM activities and the contents of the baseline are developed and made available to affected groups and individuals (e.g., the SCCB and the project leader). Software baseline audits are conducted according to a documented procedure.

Definition of Software Maintenance Inconsistent With Federal Guidelines

HUD's System Development Methodology (SDM) defines maintenance as the necessary activities performed to keep the system operational and responsive after it is placed into production. The system is monitored and post-implementation reviews are done to ensure that it is performing at an acceptable level. Corrections, deletions, modifications, and enhancements are performed on the system's hardware and software whenever it falls below the level of acceptable performance or when the level of acceptable performance is amended. IT's definition of software maintenance has not been formalized, and IT officials agree that this lack of formal definition leads to inconsistencies. IT's definition provides very little information about the sources of changes to the system.

FIPS PUB 106 identifies the three following categories for monitoring and controlling software maintenance activities: corrective[7], adaptive[8], or perfective[9]. While specific terminology may vary, classifying software changes into the three FIPS categories would facilitate meaningful analysis of maintenance data and focus on areas needing attention. For example, "corrective" maintenance customarily accounts for 20 percent of software maintenance and encompasses design, logic, and coding errors. Consequently, system managers noticing a 40 percent corrective maintenance rate would be alerted to existing or potential problems regarding program design flaws or coding errors and/or inadequate testing or review of changes. In general, IT has classified maintenance as making corrective changes.

None of the seven application systems reviewed distinguished proposed changes as "adaptive." Adaptive changes include changes in laws and regulations, as well as hardware configuration and system software configuration. Adaptive changes are often beyond the control of the software

maintainer because they must be made in order to adapt to the changes in the system's environment. However, in practice these external changes are seldom accomplished without extensive coordination. A manager who can demonstrate the frequency of external changes, supported by accurate cost and schedule data and an assessment of the impact on service delivery, may be able to influence the timing of the change or obtain additional resources necessary to meet the schedules.

Changes, insertions, deletions, modifications, extensions, and enhancements which are made to the system to meet changing user needs represent the evolution of the information system. They generally make the system more responsive to the way users do their work. They are generally discretionary in nature. In times of budget cuts, when choices must be made, corrective and adaptive changes frequently cannot be deferred because the system cannot function properly without such changes. However, perfective changes or enhancements can be considered discretionary and deferred.

The European Platform for Software Maintenance (EPSOM) presented a generic software maintenance process (Figure 4) to the Conference on Software Maintenance in 1992. We include this generic software maintenance process in Appendix E as a well-defined reference model for HUD's definition of its own software maintenance process. We view most of the activities in this model as shared activities, with IT providing the technical input to the activity and a user change control board making the decision to proceed to the next activity.

In order to categorize the various kinds of activities performed as "software maintenance" and to enable a deeper analysis of the various tasks carried out in the software maintenance process, EPSOM has adopted a more precise classification of maintenance activities--effectively specialized instances of the three FIPS PUB 106 classifications. For example, EPSOM defines *anticipative maintenance* as those changes made to software in anticipation of a problem, e.g., changes made now to adapt the software for the potential disaster associated with the year 2000, making anticipative maintenance a specialized instance of adaptive maintenance. A possible confusion is that EPSOM uses the term *evolutive maintenance* to describe changes to meet the evolving requirements of the users, corresponding to the FIPS PUB 106 classification *perfective maintenance*. They then use the term *perfective maintenance* to refer to non-functional changes, e.g. response time, execution time, memory size, etc.

If managers cannot identify changes as corrective, adaptive, or perfective, they cannot make informed decisions on software maintenance priorities. Based on the current types of classifications used to distinguish requests, HUD system managers cannot identify trends for budget purposes, or point out weaknesses in the change control process. In addition, management lacks insight regarding the true nature of the proposed change (adaptive, perfective, or corrective) and cannot evaluate the stability of the application system or the need for replacement. Without proper classification, the following types of problems can remain undetected under current circumstances:

- Weaknesses in established review and testing procedures;
- Poor contractor's performance resulting in rework of continuing problem;
- Degrading software performance; or
- Excessive maintenance costs for a particular type of software change.

Single Change Request Form Needed

During the interviews with the project leaders, we found that although numerous change request forms exist, the user community prefers to initiate changes by telephone call, or cc:Mail. Project leaders sometimes must insist that users call the User Assistance Branch to open a PTAR. However, some project leaders initiate system changes based on the telephoned request from the user.

FIPS PUB 106 states that a formal, well-defined mechanism must exist for initiating requests for change or enhancement of a system. All changes should be formally requested in writing and submitted on a standardized request form. The decision and reasons for acceptance/rejection of a change request also should be recorded and included in the permanent documentation for the system. A standardized form ensures adequate information is available for classifying, reviewing, and processing change requests. A standardized form can also be used for measuring maintenance activities.

HUD Handbook 2400.15 defines the following request forms to initiate software changes:

- Advanced Requirements Notice (ARN) initiated by the user to initiate enhancement of existing systems due to:
  - Legislative and/or regulatory developments (adaptive maintenance); or
  - User initiative (perfective maintenance);
- User Assistance Request (UAR) initiated by the user notifying the Computer Systems Group (CSG) that a production problem has occurred (corrective maintenance);
- Difficulty Report (DR) initiated by CSG for reporting execution errors (corrective maintenance); and
- Teleprocessing Assistance Request (TAR) notifying CSG that the user has encountered a teleprocessing problem and requests technical assistance.

Note that while the UAR, DR, and TAR have been eliminated by IT they still exist as official standards for documenting problems, errors, and requests for enhancement. As reflected in Table II, two of these forms were being used during the period of our review.

Additionally, the Project and Resource Management System (PARMS) defines the following tracking numbers which are used in lieu of a request form:

- A job control number (JCN) is provided to the support branch to identify work efforts not initiated by an ARN; and
- A project control number (PCN) is assigned to a work effort for tracking in PARMS.

Our analysis of Computer Management Division (CMD) Release Requests, see Table II, shows that project leaders cite a wide variety of request types to initiate the release of software changes into production. Forty-five percent of

the change requests used the three request forms defined in HUD Handbook 2400.15. Another 17 percent are based on the PTARS system. However, the remaining 38 percent do not conform to any of these request types.

If standardized change request forms are not used to record proposed changes, sufficient data may not be available to adequately evaluate the nature of a proposed software change or its resulting benefits, cost, or impact on the application system. Approved changes could be misclassified or ranked inappropriately due to insufficient information. In addition, valuable time and resources could be wasted by management in trying to discern the nature of the requested change or returning a request for further information.

Project Tracking Not Based on Measurement

> *When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind.*
>
> *- Lord Kelvin*

Performance Measurement Indicators Must Be Established

System owners have not established performance measures for software changes to aid in determining how well their application systems are being maintained. IT collects performance data in the PTARS. The system is used for recording problems, assigning problems for resolution, tracking progress on problem resolution, and assisting analysis of problems. IT uses this system to prepare the following workload and performance indicator reports monthly:

- Number of problem tickets recorded for the month, displayed by Headquarters and Regions;
- Historical analysis showing the number of days required to close a problem ticket, shown by month for a year;
- A frequency analysis of problem types (41 service requested codes) and solutions (39 service performed codes); and

- An analysis of equipment failure rates.

However, PTARS produces operational reports providing limited information on the effects of software changes. PTARS has no data on the measurable indicators such as the number of software problems, changes, types of changes, number of changes to each module, etc., that can be used to quantify the outcome of the change control process. Also, although various types of historical data were recorded in change control logs, discussions with IT application project leaders disclosed that the collected data were not used for measuring performance in software changes.

Performance measures must be established in any effort, including software development and maintenance, in order to evaluate the outcome. Managers make decisions on the basis of the best information available. A complete management system includes a performance monitoring feedback loop with successive cycles of goal setting, performance monitoring, and regular reporting. Such a system requires regular, efficient information collection, analysis, and review. A performance measurement system simply formalizes, makes more efficient, and makes explicit the decision-making process managers use intuitively. Such a system forces managers to confront hard evidence about program efficiency and effectiveness.

Without defined measurable performance indicators, system owners and IT personnel who provide technical support do not have the necessary management tools to: (1) evaluate the effectiveness of their change control process; (2) assess the overall stability of the application system; or (3) draw conclusions as to how resources should best be utilized. Consequently, system owners and IT system managers do not have the benefit of valuable information for making decisions.

As such, software maintenance for HUD's numerous application systems may be managed in an inconsistent and erratic manner. Under current procedures, unapproved modifications, unintentional, malicious and/or fraudulent insertions of source code, or poorly structured or inadequately tested program code can proceed undetected through the change process. (See subsection Test Plans And Analysis Of Test Results Are Not Routinely Prepared).

Most measurement programs use defect counts. A defect is something wrong that needs to be fixed. Bugs and problems are recorded in a tracking system to make sure they do not get lost or forgotten. Whenever a problem is discovered,

a problem record is "opened" and basic data about the problem (i.e., activity or phase, symptom, suspected cause, and type or classification) is recorded. After the fix is "approved," a change is prepared and tested, and the problem can be "closed." Upon closing, additional data is usually entered into the record (i.e., actual cause, source of the problem, and effort to fix). Appendix D provides additional discussion of software measurement.

Analysis of defects can lead to improvements to the software process. Such improvements can contribute to higher customer satisfaction, fewer emergency repairs to the software, and higher software support productivity. Classifying defects into the following categories points to specific parts of the development process and makes possible an analysis of defect-removal efficiency at each stage of the system lifecycle:

- ***Category 1*** defects include those detected by computer operations; typically these include abnormal terminations (ABENDS in the mainframe environment) and out-of-balance conditions.

- ***Category 2*** defects include those reported by a customer after a system is used in production. These include all failures to comply with requirements. Examples are improper functionality, computational errors, and failures to meet performance constraints.
- ***Category 3*** defects are those detected by the developers during the testing phases of the system lifecycle. This cycle is bounded by the release of the software into production usage.

- ***Category 4*** defects include those detected by the developers prior to testing. These are typically detected in quality assurance activities such as reviews, inspections, and walkthroughs.

Causal analysis of defects leads to improvements in the development, testing, and inspection processes. Defects in categories 1 and 2 reflect the overall failure of the System Development Methodology to detect and remove significant defects prior to release. Defects in category 3 reflect the failure of quality assurance processes prior to testing. Defects in category 4 reflect failures in the early phases of development such as unclear user specifications.

Process improvements are achieved when management uses its knowledge of defect causes to change processes. Category 1 and 2 defects should lead to

intense management scrutiny of the System Development Methodology and its project management for the project or system experiencing these defects. Category 3 errors should lead to a more active quality assurance process to detect defects early. Category 4 defects should lead to preventive measures designed to reduce the introduction of defects, e.g., new analysis tools or more user involvement in the specifications.

Testing to Control Quality Must Be Strengthened

Test Plans And Analysis Of Test Results Are Not Routinely Prepared

During our review, only two of the seven IT project leaders provided us with a Test Plan and a Test Analysis and Results Report. Only two of the project leaders indicated that test results were reviewed and approved by the user before a change was put into production. Most of this documentation does not define how testing will be conducted or what are the expected results. The other IT project leaders were unable to provide documentation that testing of software changes was conducted and evaluated during our review period.

Using the IT Quality Assurance Staff analysis, we found the following indicators of insufficient testing:

- Sixteen percent of all releases resulted in a problem;
- Twenty-seven percent of large releases resulted in one or more problems (nine percent of all releases were large releases, meaning that they had more than 24 release elements); and
- HDS, DB2, IEF, and MAPPER releases have a higher rate of problems than the average of all releases.

Although 84 percent of the software releases did not result in production problems, 16 percent of all releases created new problems that were costly in terms of IT and program office staff impacts. Specifically, 26 releases resulted in 44 re-releases to correct one or more problems with the previous release. This implies that emergency corrections had to be made to software changes that failed to work properly after having been released into production operation. Each software release was tested and approved by IT change management

procedures for release into production operation. These statistics support a conclusion that established review and testing procedures are inadequate, and perhaps contractor performance could be improved.

QMS also reported that 51 percent of all release requests indicated that the release was intended to correct a problem. These problem correction rates are much higher than defined as normal by FIPS PUB 106, suggesting information systems that are not stable or reliable, and should be evaluated for replacement. However, as we have noted, IT does not define and classify software maintenance in accordance with FIPS PUB 106. Therefore, it is likely that some percentage of these releases classified as corrections would have been classified as adaptive maintenance if FIPS PUB 106 definitions had been used. However, the 51 percent of all release requests for which the release was intended to correct a problem is still higher than FIPS estimates for corrective and adaptive maintenance combined. This supports the need for accurate classification, measurement, and causal analysis discussed in the previous section.

Testing is a critical component of software maintenance. Testing procedures promote software maintainability, quality, and system integrity. As such, the test procedures must be consistent and based on sound principles. If project leaders do not document the strategy and limitations of testing, changes to software may not be tested sufficiently to account for all valid, invalid, expected, and unexpected outputs. The contractor executing the tests may not recognize the relevant limitations on the test because of the test conditions. Test results will then be interpreted as conclusive when, in fact, they do not represent what will happen in production environment. Similarly, not all functions of proposed software changes may be tested. Insufficient testing and analysis of test results might result in source code that fails when introduced to the production environment, due to unforeseen transaction conditions, interfaces, or user input.

During the test stage, the software and its related documentation should be evaluated for production readiness and implementation. The goal of testing is to find errors and/or performance problems, and, therefore, a test plan should: (1) define the degree and depth of testing to be completed; (2) describe the expected results; and (3) test for valid, invalid, expected, and unexpected cases. Federal guidelines outline the format and content of test plans and test analysis reports, and emphasize the importance of:

- Identifying and segregating the various functions of the program to be tested;

- Describing the strategy and limitations of the testing; and

- Describing the source data and expected result data for each planned test.

Lack Of Verification And Validation Testing

HUD's SDM does not require software verification and validation[10] (V&V) testing during the software maintenance phase. V&V testing is described in the SDM as unit, integration, and systems tests to be conducted by the developers and the users during the Design Phase. None of the project leaders for the seven application systems we reviewed required V&V testing of software changes.

Also, HUD's V&V testing does not follow the intent of FIPS PUB 132 which requires V&V testing be conducted independently of the system development testing. Even if the software was not verified and validated when originally developed, under FIPS Pub 132 a new V&V plan should be developed to test modifications made during the lifecycle's maintenance phase.

V&V testing should be done for both critical and non-critical application systems. V&V uses a structured approach to analyze and test the software against all system functions and all hardware, system users, and other software interfaces. Through V&V testing, high risk errors are detected early, software performance is improved, and confidence is established in software reliability. The cost of conducting independent V&V is offset by cost advantages of early error detection and improved software reliability and quality.

Without independent V&V testing, system management lacks objective assurances regarding the content of application source code, the completeness and usefulness of system and user documentation, the interaction of application system baseline components, and the stability of system operations. Consequently, any of the following effects can happen:

- Software changes are placed into production without the completion of relevant system and user documentation;

- Unapproved software changes are introduced into the production environment; or

- Important testing, review or approval procedures are omitted from the maintenance cycle inadvertently or intentionally, in an attempt to expedite the modification process.

These effects can result in application software that is difficult to maintain because of insufficient or outdated system documentation. Also, the adequacy of user documentation may be compromised. This may affect the training of new employees or the performance of established application users. In an extreme case, malicious or fraudulent code may be introduced without management's knowledge, and the performance of the application system may be severely jeopardized.

Departmentwide Software Configuration Management Process Needed

Configuration Management Is Informal and Optional

We found that none of the seven application systems used a software configuration management (SCM) system (e.g., ENDEVOR, which HUD has purchased but implemented for only one application) to: (1) do impact analysis of changes; (2) require audit trails for emergency software changes; (3) create automated approval mechanisms that can be customized to the change management process, (4) control migration between production and development libraries, and (5) provide an inventory of the baseline copy of each program with a record of the changes that have been made to the product. Our review also disclosed that several of the IT project leaders do not maintain a configuration inventory as required by the SDM. Instead, these project leaders rely on the CMD Release Request form for configuration inventory and configuration status reporting.

Each application system uses some type of controlled access to source code through library management tools[11] (e.g., Interactive Processing Facility (IPF) or Librarian), and a release procedure is used to prepare changes for transfer to the production environment (e.g., Software Migration System (SMS) or Program Migration System (PMS)). Although library management tools can track successive versions of individual software programs, such utilities cannot recognize which version of each software component comprised a specific prior

production release of the application. Typically configuration management products control the release of a program from a production library to a development library to permit a change and control the approval process until the program is moved back into the production library.

However, the need for configuration management is left to the discretion of the individual project leaders. One project leader, who currently uses the PMS tool, plans to migrate to the ENDEVOR product because it is a more powerful configuration management tool than PMS. Another project leader wants to implement the Life Cycle Management (LCM) software from Realia. LCM controls programs in a PC-based development environment, and provides check-out/check-in capabilities, concurrent development controls, and migration to the mainframe controls. The project leader for the A67 LOCCS system has developed rigorous controls over the program source library using the UNISYS IPF. These controls require that a program be checked out of the source library for review, and maintains a log of who checked the module out, what library it was placed into, and the change delta when it is returned. These controls contain many of the features of a configuration management software package.

The need for configuration management and version control may be acute. Our analysis of the release requests shows that many software releases may be required to complete a single change request, and that the same program may be changed and released into production several times to complete a single change request (see Table III and Figure 5 below.)

NOTE: Figure 5 shows multiple releases for the same change request. Eight change requests or PTARS account for 36 (41 percent) of the 88 releases made for F72 TIIS during the evaluation period. Another 32 releases (36 percent) are associated with miscellaneous or unknown change requests. Only 23 of the change requests were completed with 20 (23 percent) single releases, and 3 of the 23 were paired with another change into a single release.

Not only were there multiple releases to complete the change request, the same program could have been released several times to complete the request. This is shown in Table III.

The GSA *Guide for Acquiring Software Development Services* defines software configuration as an arrangement of software parts, including all elements necessary for the software to work. Configuration management refers to the process of identifying and documenting the configuration and then systematically controlling changes to it to maintain its integrity and to trace changes. Since there are always multiple versions of application software, it is

very important to be able to identify which version of a module is associated with a particular program configuration. Version control allows program developers and maintainers to locate the latest version of a program accurately, reliably, and consistently. Version control also enables system managers to go back to prior configurations of an application software should a newly modified version fail to operate correctly after being placed into production.

Without software configuration management tools, system managers would find it difficult to identify which version of numerous software components comprised a specific operational configuration. Similarly, unrecorded "emergency" changes can happen and affect numerous other components of the application system. Considering the complexity of many of HUD's application systems, it could be costly and time consuming to accurately identify and reinstate a specific prior software configuration for use. Instead, system managers would most likely be forced to live with the current faulty version of the system and to correct any errors or deficiencies until the application software was functionally corrected.

Constant revisions to a baseline configuration as shown in Figure 5 could have significant repercussions. Even a single modification could impact numerous components of the baseline configuration and, in some cases, add components to the configuration. Many of the application systems reviewed implemented hundreds of modifications during a single calendar year. Therefore, manually performed impact analysis, for proposed software changes, may be insufficient to identify and evaluate the effect of each modification on other components of the baseline configuration. Without adequate impact analysis, unforeseen problems during implementation could arise because of software component interactions that were overlooked during software testing. Inadequate or incomplete impact analysis could lead to production failure in extreme cases.

However, use of a SCM product would ensure version control for application systems that are constantly undergoing modifications. Also, the implementation of a product, such as ENDEVOR, would not interfere with library management systems already in use by several applications. With ENDEVOR, system managers could "rollback" to a prior version of the system with assurance that all system components would be synchronized to perform properly.

Key Success Factors In Controlling Software Changes

System Owners Must Take Responsibility to Manage Software Changes Made To Their Applications

OMB Circular A-130 states that basic management controls for agency information systems are fundamental to sound information resources management. HUD managers, as system owners, depend upon application systems to carry out their programs and are accountable for the performance of the system. However, program officials are not managing the software change process. System owners are not involved in the entire software change process. While they often initiate the change request, they do not systematically evaluate the impact the changes have on the quality of the software.

The change control deficiencies we noted are attributable to the lack of change control boards within the system owners' organizations. FIPS PUB 106 stresses the importance of a centralized approval point for change requests. A centralized approval process enables one person or group of persons to have knowledge of all the requested and actual work being done on the system(s). This is done to prevent problems caused by two or more independent changes to the system that will be in conflict with each other. Another reason is to coordinate similar requests so that details can be combined and the total amount of resources required can be reduced. A centralized change approval board in the system owners' organizations affirms the system owners accountability and responsibility for managing their application systems and accountability for the performance of these systems.

HUD's SDM Does Not Provide Adequate Direction For The Software Change Control Process

HUD's SDM defines maintenance as the necessary activities performed to keep the system operational and responsive after it is placed into production. The system is monitored and post-implementation reviews are done to ensure that it is performing at an acceptable level. Corrections, deletions, modifications, and enhancements are performed on the system's hardware and software whenever it falls below the level of acceptable performance or when the level of acceptable performance is amended.

However, the SDM lacks procedures to implement FIPS PUB 106 guidelines on controlling and improving the software maintenance process, including using effective techniques and tools. Specifically, the SDM does not address the following key management issues that would establish controls over software

maintenance:

- A process which implements FIPS PUB 38 guidelines to define the format and content of test plans and test analysis;
- Software configuration management and version control techniques to manage the maintenance process within an evolving and dynamic application system;
- Identification of benefits of software configuration management tools or support the use of an automated tool to ensure an adequate audit trail of system modifications;
- Descriptions of how change request and problem reporting data can be used to evaluate the adequacy of current maintenance practices, identify questionable trends in software maintenance, or evaluate application stability;
- Identification of what types of maintenance data provide the most reliable and useful metrics information, how data should be measured, or how management can interpret measurements to improve their control of application software maintenance processing;
- Definition of the quality assurance functions to manage the maintenance process;
- Identification of the types of testing that should be mandatory for evaluating the anticipated performance of a software change; and
- Specifics on the formulation of software verification and validation plans (SVVPs) and independent V&V testing for software maintenance activities, whether or not the initial application development products were subjected to V&V testing under existing Federal standards.

FIPS PUB 106 Does Not Define Software Maintenance Management Responsibilities For System Owners and Technical Support Organizations

FIPS PUB 106 recognizes the importance of management in the software maintenance process, stating that management is clearly one of the most important factors in improving the software maintenance process. It states that management must examine how the software is maintained, exercise control over the process, and ensure that effective software maintenance techniques and tools are employed. In addition, software maintenance managers are responsible for making decisions regarding the performance of software maintenance, assigning priorities to the requested work, estimating the level of effort for a task, tracking the progress of the work, and assuring adherence to system standards in all phases of the maintenance effort.

This guidance should place the same emphasis on program management accountability and technical support organization fiduciary responsibility to their program clients that is found in OMB Circular A-130. However, FIPS PUB 106--which is the only Federal guidance on software maintenance--assumes that all managers will know how to fulfill these responsibilities. It does not provide guidance about the effective techniques and tools that managers must employ. It does not define techniques and tools that aid the exercise of control over the software maintenance process. It does not define techniques and tools that aid in estimating the level of effort for a task. Agencies are left to exercise these technical responsibilities without adequate guidance.

The deficiencies in FIPS PUB 106 guidance are being addressed in the PCIE consolidated report to OMB. Nevertheless, it is still incumbent on individual Federal agencies to establish their own guidance in the absence of Federal guidance.

Project Management Personnel Rely Heavily On Adequate Performance Of Supporting Contractors

IT project management personnel rely extensively on an adequate and thorough performance by supporting contractors who perform the design, coding, and testing of approved changes. We found little evidence that either the contractors' technical performance or the adequacy of the modified source code were sufficiently reviewed by responsible IT project leaders prior to

implementation. Quality review of contractors' performance in change control is essential to minimize rework costs and risks of errors and system failures (See Chapter 4).

Recommendations

Acting Director of Information Technology, and The Technology Investment Board (TIB)

3 (a) Issue guidelines with the concurrence of the Management Committee for system owner organizations to use in managing the software maintenance of their applications in accordance with OMB Circular A-130:

- Establishing Change Control Boards;
- Evaluating the systems usability in supporting organizational goals and mission.

3 (b) Define software maintenance and classify changes according to the functional type of change being made to the application systems (e.g., adaptive, corrective, perfective).

3 (c) Identify the measurements needed to support Departmentwide management of information technology. The measurements shall include:

- Resource tracking - quantification in dollar amounts of resources used as the input for production of a service or product (i.e., estimating and tracking resource use, tasks, deliverables, and milestones);
- Work product tracking -
  - The number of units of the product or service provided to the customer, and appropriate measures of size and complexity (e.g., function points, cyclomatic complexity, Halstead code measurements, Kiviat diagrams); (see Appendix C on software measurements),

- Tracking and control of source code, test case, and document versions;

- Quality tracking -

  - Tracking and control of problems, defects, and open issues,

  - The level of service or product quality, both in terms of customer satisfaction (external quality) and of work performed to provide the service (internal process quality); and

- Change tracking - a common tracking system that will track all software changes made to each application system, regardless of the type of proposed change or its anticipated level of effort.

3 (d) Adopt a standardized form for initiating all requests for software changes, regardless of anticipated level of effort. The form shall support the measurement program and minimally include: requestor name, date, priority, problem description and justification, type of change, management approval, and completion date.

TIB Representatives and System Owner Organizations

3 (e) Follow the guidelines issued by the Management Committee to establish a centralized change control board (See Recommendation 3 (a) above). Ideally, the Change Control Board should be chaired by a senior manager responsible for operations.

3 (f) Follow the guidelines issued by the Management Committee to periodically evaluate the application systems' capability to support Departmental and office strategic objectives (See Recommendation 3 (a) above). The evaluation should include a historical review of changes and problem data similar to the IT study of release management so managers can identify which software modules require a high amount of corrective changes, rework by contractors,

and rework due to testing and review problems. The review should also determine whether it would be more cost effective to redesign than continue maintaining these modules.

3 (g) Follow the guidelines issued by the TIB for conducting user acceptance testing and reviewing the test results of software changes. At a minimum, the appointed reviewers shall review the results of the user acceptance tests (see Recommendation 3 (k) below) to determine if end users are satisfied with the changes and that the system will meet the needs as intended.

Acting Director, Office of Information Technology

3 (h) Modify and/or establish policy documents to include an organizationwide software maintenance policy describing in broad terms the responsibilities, authorities, functions, and operations of both system owners and IT;

3 (i) Establish a policy for using an integrated software configuration management (SCM) tool, such as ENDEVOR, for all application systems on all platforms (i.e., UNISYS, HDS, LAN, PC). In preparation for adoption of this recommendation, HUD management shall establish a definite implementation schedule covering each of the affected systems. The schedule should be aimed at full SCM implementation within a short period of time.

3 (j) Establish policies for software V&V plans, as defined in FIPS Publication 132. Independent V&V testing shall be performed for significant adaptive and/or perfective software changes within the application system. Although corrective modifications shall not require V&V testing, they shall be tracked, and evaluated in subsequent V&V testing of adaptive or perfective changes to avoid potential conflicts. The policy shall:

- Stipulate uniform and minimum requirements for the format and content of developed Software Verification and Validation Plans (SVVP), with graduated requirements based on level of effort thresholds;

- State purpose and benefits of independent V&V testing;

- Describe, in sufficient detail, the required inputs and outputs to be included in a SVVP;

- Suggest optional V&V tasks to be used to tailor a SVVP, as appropriate, to a particular V&V effort; and

- Ensure that the operational system uses an optimum, least-cost mix of resources to meet user functional, data, and other systems' compatibility requirements.

3 (k) Modify the SDM to include the following:

- Detailed source coding standards applicable to all phases in the SDM that include, but are not limited to:

  - standard data definitions approved by Central Information Management,

  - coding conventions,

  - structured, modular software,

  - single high-order language,

  - well-commented code, and

  - compiler extensions.

  (Note: The above standards establish measures to facilitate management review.)

- Formal testing procedures for software modifications that include tests for all valid, invalid, expected, and unexpected outputs during maintenance. Require acceptance testing by the user community for software changes that, at a minimum:

  - represent a new program or module within the

application,

- represent a major system enhancement to the application,

- represent a level of effort that is technically considered by management as a "development" (perfective) project, than a routine or minor maintenance action item, or

- are a group of software changes that collectively represent a considerable change to the application's performance.

- An objective review of modified source code and test documentation to ensure the modified code conforms with established source code standards and effectively complies with the change request, and that no additional unapproved changes were introduced by the programmers during the coding process. This review must be performed by an individual or group of individuals with sufficient technical knowledge to detect programming deviations.

Auditee Comments and OIG Response

IT has provided substantial comments to this finding, including access to project files. IT management has generally agreed with the recommendations but not with the way that we have portrayed each condition. Our evalualtion of IT comments is discussed in detail in Appendix A.

Chapter 4

HUD Has Not Been Using Performance-Based Contracting Methods for Software Development and Maintenance Contracts

The Offices of Procurement and Contracts (OPC) and Information Technology (IT) are not using performance-based contracting methods to award software development and maintenance contracts[12] as required by Federal Procurement Regulations. OPC has consistently awarded Cost-Plus-Fixed-Fee (CPFF) contracts rather than incentive type of contracts for software development and maintenance. We found that the Statements of Work (SOW's) prepared for eight contracts did not include measurable performance standards, acceptable quality levels, and quality assurance surveillance plans. Consequently, these eight contracts, worth over $135 million, of which approximately $8.6 million were billed for the seven applications during our review period, do not provide the incentive or accountability for contractors to perform well and control costs.

Also, without performance measures, HUD cannot determine whether the millions of dollars paid to contractors each year for software services are worthwhile. Moreover, without performance measures, HUD cannot hold contractors accountable for deficient performance that could cause serious operational problems.

According to OPC and IT, the inability of program officials to define software requirements, and the need for flexibility in responding to time sensitive legislative and administrative requests are the reasons for not using performance-based contracting.

Federal Requirements on Performance-Based Contracting

The Office of Federal Procurement Policy (OFPP) Letter #91-2 entitled *Service Contracting*, dated April 9, 1991, requires agencies to use performance-based contracting methods when developing Statements of Work (SOWs); develop formal, measurable performance standards and surveillance plans for assessing contractor performance; and use contract types that motivate contractors to perform at optimal levels. The letter emphasizes the use of performance requirements and quality standards in defining contract requirements, source selection, and quality assurance. Furthermore, to the maximum extent

practicable, contracts shall include incentive provisions to ensure that contractors are rewarded for good performance and quality assurance deduction schedules to discourage unsatisfactory performance. These provisions are based on measurement against predetermined performance standards and surveillance plans. Agencies must document in their contract files the reasons for those instances where performance-based contracting methods are not used.

OFPP Pamphlet #4, entitled *A Guide for Writing and Administering Performance Statement of Work for Service Contracts*, dated October 1980, provides guidelines for writing and administering performance statements of work for service contracts. It describes a systematic means to develop statements of work and quality assurance surveillance plans in order for agencies to define and measure the quality of contractors' performance.

Elements of A Quality Assurance Surveillance Plan and the Performance Requirements Summary

Both the OFPP Letter #91-2 and OFPP Pamphlet #4 identify the Quality Assurance Surveillance Plan (QASP) and the Performance Requirement Summary (PRS) as the two primary documents agencies can use for planning, measuring, and monitoring contractor performance.

A QASP is a well-defined written document used to ensure that systematic quality assurance methods are used. It is developed as an extension to, and included within the contract statement of work. Contractors use the QASP to develop their written quality control program. The QASP's primary function is to determine whether the services provided by the contractor meet the Government's quality performance standards using systematic quality assurance methods. A QASP has four performance elements; performance standards, acceptable quality levels, performance indicators, and performance values. Performance standards and acceptable quality levels are to be developed concurrently with the QASP and clearly defined in the SOW.

Performance standards are characterized as an acknowledged measure for comparison of quantitative and/or qualitative values. These standards are typically derived from industry directives and governmental and agency standards. It is important to select performance standards that pertain to the deliverables, products, or services being acquired. Also, each performance standard used should have defined acceptable quality levels.

Acceptable Quality Levels (AQLs) are the maximum percentage of allowable variance from the norm before a deliverable, product, or service is rejected. The AQLs recognize that defective performance sometimes happens unintentionally.

Performance indicators are characteristics of an output that can be measured. It may measure quantity as well as quality with the emphasis on using quantitative measurements whenever possible. Usually, the measures are stated as rates, i.e., a means of expressing something as it related to a fixed amount of something else. A person must be careful to choose performance indicators that are realistic for the service performed. Whenever possible, one must choose an indicator that measures the service by a numerical value.

Performance values are a composite of performance standards and acceptable quality levels, which describe the quality of the service that can be measured (performance indicators). Performance values enable a person to place realistic demands on the contractor performance. Performance values permit the writing of the performance-oriented SOW and eases the development of the QASP. These outputs are evaluated periodically by the quality assurance evaluator (QAE) whose ultimate responsibility includes developing an activity's schedule based on requirements within a surveillance plan.

Once these performance elements are known, they are categorized into a document called Performance Requirements Summary (PRS). The PRS summarizes the performance requirements of the contractor's evaluation. This document is an attached exhibit to the SOW. This summary identifies the performance indicators related to each service; defines the required standard for each service; states the acceptable quality levels associated with each standard and service; identifies the surveillance method to be used; and identifies the deduction percentage from the contract price for exceeding the acceptable quality level.

An Example of a Measurable Performance Standard, Acceptable Quality Level, and Performance Indicator

To use an illustration, under the performance measurement, "Quality of Project Performance" in the ADP solicitation referenced in the auditee response to this draft finding, one measurement indicated that the contractor "will be evaluated to the extent the work was error free." However, a more appropriate measurable performance standard here could be **100% of the software changes moved into the**

production environment do not require a correction release . The associated Acceptable Quality Level would indicate that **deviation of more than 5 percent** (target based on current experience of 16 percent as noted in Chapter 3) **of releases will result in a deduction** . The performance indicator would be **operational in the production environment without error for two or more production cycles**.

HUD's Software Development and Maintenance Contracts Are Not Performance-Based

We reviewed eight ADP software and maintenance contracts out of a total of 71 ADP agencywide contracts. The contract numbers are HC 16325, HC 16326, HC 14747, HC 16334, HC 16313, HC 16315, HC 16311, and HC 16310. These contracts provide software development and maintenance services to the entire agency. They were selected based on the contract work performed on the seven systems from October 1, 1992 through March 31, 1994. The contracts' negotiated value at full performance is in excess of $135 million depending on funding availability. At the time of our review, from the award date to March 31, 1994, these contracts were funded in excess of $70 million. Also, during our review, approximately $8.6 million were billed for the seven systems.

Performance-based contracting methods consist of SOWs that describe "what is to be performed" and surveillance plans that ensure systematic quality assurance methods are used to facilitate the assessment of contractor performance. HUD's Statements of Work (SOWs) for the eight ADP software development and maintenance contracts do not include performance standards and acceptable quality levels. HUD is not using quality assurance surveillance plans, to include performance requirements summaries, as a means to monitor the contractor's performance. In addition "Task Specifications," which are prepared to supplement the SOW, generally contain vague requirements without specifying performance standards or acceptable quality levels.

Statements of Work Lack Measurable Performance Requirements

We examined the SOWs and the preaward documentation for the eight software development and maintenance contracts to determine whether: (i) the contract SOWs contained measurable performance standards, acceptable quality levels,

and performance values; (ii) any follow-on contracts contained more definitive SOWs and performance standards; and (iii) the contract files contained the appropriate justification for those contracts that did not use performance-based contracting methods.

We found that none of the eight contract files contained the required documentation to justify why performance-based contracting methods were not used. We also noted that all eight of the contracts contained identical "boilerplate" SOWs except for the staffing requirements section. The SOWs for the eight contracts we reviewed outlined seven general criteria for evaluating contractor's performance: efficiency, ingenuity, responsiveness, perceptiveness, thoroughness, timeliness, and resourcefulness. The criteria are vague and subjective. They cannot be linked to any specific measurable performance standards and acceptable quality levels. For instance, factors such as reliability, schedules, budgets, user satisfaction, etc., are not measured.

OPC and IT officials explained that "boilerplate" SOWs are used because they allow flexibility in responding to time- sensitive legislative and administrative requirements. However, SOWs for time sensitive tasks require specific, not "boilerplate," performance and quality requirements to ensure successful timely completion of tasks with minimal risk of rework.

Another reason given for using "boilerplate" SOWs is that program officials cannot always specify software requirements. This is not an acceptable reason. The SOW is a most important element of a contract. It should both clearly communicate to the contractor the Government's expectations and serve as a basis for evaluating the contractor's performance. To avoid developing vague and imprecise SOWs, measurable performance standards, acceptable quality levels, and related documents must be identified within the SOW for the individual contracts. This gives HUD the necessary leverage to ensure that contractors' performance levels are of an acceptable quality and that HUD only pays for products and services that meet the contracting standards.

Without performance measurements, HUD cannot determine whether the millions of dollars paid each year for contractors to develop and maintain application systems are spent wisely. Further, if performance data are not collected and analyzed, HUD cannot prevent serious operational problems caused by deficient contractor performance.

Quality Assurance Surveillance Plan and Performance Requirements Summary Are Not Prepared

Quality Assurance Surveillance Plans and Performance Requirements Summaries have not been developed and incorporated as part of the SOWs for the eight contracts reviewed. We brought this issue to the attention of HUD officials responsible for evaluating contractors' performance. These officials indicated that they do not quantify the seven performance criteria of efficiency, ingenuity, responsiveness, perceptiveness, thoroughness, timeliness, and resourcefulness stated in the SOWs. Instead they rely on their personal and professional expertise and user acceptance in determining the acceptability of deliverables and for measuring contractor performance. However, without establishing and incorporating measurable performance standards, acceptable quality levels, and quality assurance documents in the SOW, HUD cannot hold the contractor accountable for the quality of products or services provided. As a result, HUD lacks assurances that the contract work is cost effective and produces the intended results.

Task Specifications Lack Important Performance Standards And Are Not Prepared According to Internal Instructions

To meet management's need for flexibility, OPC allows IT to use an agency-unique contracting tool entitled "Task Specification." According to OPC's office instruction "90-5" dated June 6, 1990, a task specification is a contracting tool that extends the GTR's authority over the technical direction of the contract. OPC does not review these Task Specifications for completeness and conformance with the applicable contract requirements.

IT's office instructions on preparing Task Specifications contain requirements that can be measured. IT-CON-04 entitled *Task Specification Preparation and Issuance* dated February 9, 1988, requires the task specification to:

- Describe in detail all the work to be completed;
- Break down the work into subtasks to be completed by a specific date;
- List the products to be delivered; and, most importantly,
- Satisfy the requirements given in the SOW of the contract.

A schedule of completion and the delivery of a satisfactory product can be used to measure the performance of a contractor. IT-CON-04 also requires all products to be accepted or rejected according to "IT-CON-05," entitled *Contract Product Acceptance Instructions*. IT-CON-05 requires acceptance criteria, or measuring elements in the contract or task specification to determine if the product meets the requirements. It also requires the acceptance or rejection of a product to be completed within 5 working days after receiving it from the contractor.

Our review of the Task Specifications for the eight contracts in our audit sample revealed that the Task Specifications were vague and did not include performance standards and acceptable quality levels (See Appendix D for an example of a Task Specification used). The Task Specifications also did not meet the preparation, issuance, and contract product acceptance requirements of internal office instructions IT-CON-04 and IT-CON-05.

We evaluated 78 task specifications which included all task specifications and related modifications under our audit sample (See Table I). This evaluation disclosed:

- Task specifications did not contain performance standards, acceptable quality levels, or product acceptance criteria;
- 94 percent described the tasks to be performed using generic language;
- 97 percent of the language used was either identical or almost identical from task specification to task specification;
- 86 percent lacked product delivery dates;
- 98 percent lacked detailed explanation for modifications of task specifications; and
- 61 percent of the task specification product acceptance forms were not completed on time.

OPC and IT need to conduct quality reviews of the task specifications to ensure that they adequately describe the work and acceptance criteria. Additionally,

both the SOWs and Task Specifications are not performance-based and do not meet either Federal procurement policy or internal office instruction requirements. As a result, IT's personnel managing software contracts cannot evaluate contractors performance to hold them accountable for the results of their work and ensure that HUD only pays for services that meet contract standards.

Contractor Performance Problems Could Exist In Software Maintenance

While we did not conduct an extensive quality assurance review of the work performed by the contractors, we did find indicators in the software change control process (See Chapter 3) that suggest quality problems exist in the software change release process that could have been avoided or mitigated with implementation of contract performance measurements. For example, an analysis of software releases for the seven systems identified in our review revealed that numerous software releases may be required to complete a requested change; and numerous emergency releases were made.

Incentive Type Contracts Are Not Used For Software Services

Federal Acquisition Regulations (FAR) 16.103 state that the contract type is negotiated to provide the contractor with the greatest incentive for efficient and economical performance. The primary factors in determining what type of contract to select depend on the ability to predict the quantity desired or to define the nature of the work to be performed.

OPC almost exclusively awards CPFF contracts for software development and maintenance efforts when other, cost-effective and performance oriented contract types could be used.

OFPP Policy Letter #91-2 requires agencies to carefully select acquisition and contract administration strategies, methods, and techniques that best accommodate the requirements. Agencies are to consider using contract types that provide performance incentives and deduction schedules. Contracts

containing award or incentive fees motivate the contractor to improve performance and reward the contractor for good performance. Deduction schedules enhance the contractor's accountability and discourage unsatisfactory performance. The use of deduction schedules does not imply that the contractor may knowingly perform defective service. It simply implies that the Government recognizes defective performance sometimes happens unintentionally and may be a result of circumstances beyond the contractor's control.

Contract types that motivate contractors to perform at optimal levels include:

- ***Cost Plus Award Fee*** - This contract type allows the contractor to earn a base fee regardless of his performance. Additionally, the award fee is based on the Government's evaluation of the contractor's performance. The terms of the criteria shall be stated in the contract;

- ***Cost Plus Incentive Fee*** - The fee is determined by comparing actual cost to target cost. The target fee is calculated by applying a fee adjustment formula (share ratio) to the difference between the target and actual cost; and

- ***Fixed Price Incentive*** - A price equal to the sum of the final negotiated cost and final profit. The final profit is determined by comparing final negotiated cost and adjusted target profit by applying a formula (share ratio).

A key factor in determining if a CPFF contract should be used is the inability of the procuring agency to sufficiently describe the work requirements. Consequently, CPFF contracts place the Government at a higher risk than other contract types because of this lack of identifiable work requirements. The use of CPFF contracts as the sole contracting vehicle minimizes the contractor's incentive to perform well and control cost. CPFF should be used only after other performance oriented contract types have been considered.

The eight ADP software development and maintenance contracts we reviewed were all CPFF contracts. HUD's excessive use of CPFF contracts has been noted in a report issued by the General Services Administration (GSA) on June 8, 1992. The report recommended that HUD define users' requirements more

clearly so that other contract types can be used. Although HUD management agreed with the recommendation, HUD continues to rely on CPFF contracts for software maintenance and development work.

HUD's Current Efforts To Implement Performance-based Contracting Fall Short of OFPP Policy and Guidelines

In response to our discussion draft finding, IT indicated that it had actively taken the lead in moving toward performance-based contracting methods for software maintenance and development contracts. In 1994, the Assistant Secretary for Administration signed a pledge with OFPP to use the performance-based contracting methods prescribed in OFPP Policy Letter 91-2 for service requirements identified in an attachment to the pledge. The service requirements identified were a contract for software development, maintenance, and information resources management, currently valued at $89.3 million and due for follow-on award in February 1995. The Department pledged that this service would be awarded based on performance work statements including measurable performance standards, surveillance plans, best value selection procedures, and fixed price contracts with positive and negative incentives.

HUD staff, working closely with GSA experts, issued two solicitations within the last 9 months as their first efforts in developing performance-based contracts. One of the two contracts was for ADP services. The five performance measures incorporated into solicitation RFP #DU100C000016347 are: (1) quality of performance, (2) timeliness, (3) use of resources, (4) completeness/quality of documentation, and (5) innovation.

Unfortunately, we found the solicitation SOW still did not meet the intent of the OFPP policy and guidelines for performance-based contracting. We saw no evidence of measurable performance standards, acceptable quality levels, or a Quality Assurance Surveillance Plan incorporated into the contract. The generic work standards used in the solicitation were identical with those used in previous ADP contracts. We concluded that the Department has not met its pledge to the Office of Federal Procurement Policy in using performance-based contracting for software development, maintenance, and information resources management.

Based on the response to our discussion draft finding, the evaluation process

will consist of: (i) performance reviews of interim deliverables so that the final deliverable can be evaluated, and (ii) an evaluation report to be completed for each deliverable. After the evaluation period, the Government Technical Representative (GTR) will summarize the results of the reviews and arrive at the level of performance and recommend the appropriate action to the Contracting Officer. However, this again is subjective in nature. For example, the performance standards and acceptable quality levels that the GTR will use to measure and evaluate the work have not been defined.

IT also indicated that CPFF contracts provide them the flexibility to respond to dynamic legislative and management requirements, besides changing customer requirements. However, these CPFF contracts are not performance based. Without establishing and incorporating measurable performance standards, acceptable quality levels, and quality assurance documents in the contract, HUD cannot hold the contractor accountable for the quality of products or services provided. As a result, HUD lacks assurances that the contract work is cost-effective.

We Believe That Establishing a Project Office Can Facilitate the Implementation of Performance-based Contracting

IT and OPC have cited the inability of program officials to define software requirements, and the need for flexibility in responding to time-sensitive legislative and administrative requests as the reasons for not using performance-based contracting. While we have not conducted the audit work necessary to make a recommendation, we believe that primary organization heads should consider establishing a project office whose principal objectives are to ensure coordination between program organizations and IT and its contractors in defining measurable requirements and specifications during system development and maintenance. This office would assist management to assess the success of these efforts.

Under OMB Circular A-130, the program manager is responsible for obtaining the information system(s) necessary to fulfill the mission. Additionally, HUD Handbook 2400.1 requires that primary organization heads and their staffs play key roles in the ADP planning and budgeting process. They provide initial requirements and specifications for ADP systems, work in close coordination with the Office of Administration during development, certify that the resultant

systems meet their needs, and participate in implementation and ongoing operational activities. The establishment of a "project office" would ensure coordination between customers and developers to develop measurable requirements and specifications.

For many years, construction and defense industries have successfully applied the concept of the "project office" as a tool to manage risk on major projects. The project office is responsible for ensuring that every reasonable step is taken to ensure the success of the project. The project office is not responsible for the development project (e.g., to deliver an operational information system). Its principal objectives are to ensure open, fact-based dialog between customer and developer regarding the processes that maximize chances for program success and to provide objective and independent assessments of estimates and milestone status.

The project office must be independent and objective. If the customer or developer can silence the project office, its effectiveness is compromised. Typically the project office reports at a level in the organization that insures access to key decision makers and the opportunity to escalate issues to the highest levels. The project office identifies industry- best practices, and champions their use within the organization. It is not an audit function or an enforcement function. The project office must have at least one senior individual with large-scale project management experience.
For example, a project office for the CFO should include both an accounting systems expert and an information systems expert. A project office for the CFO might address many of the following issues:

- Define both CFO and IT responsibilities for operating each application;

- Implement a service-level agreement with IT that recognizes IT's role as a service provider and CFO's role as the system owner.

- Measure the performance of IT and its contractors on each task they complete, for example:

    - Output measures should describe the number of units of goods or services that were produced, either by discrete definition, e.g., function points for software, or by a proxy

measure that accurately represents the product;

- Efficiency measures should be the total cost per unit of output;

- Effectiveness measures should reflect processes controlled by IT management, to include:

  - quantity of output;
  - quality of outputs as defined by CFO (production problems and ABENDS are quality failures);
  - timeliness of outputs; and
  - customer satisfaction, based on statistical survey of CFO customers;

- Productivity measures should incorporate the amount of work done, the effort devoted to it, and the time that it takes, e.g., the number of function points of software delivered each month per developer;

- Throughput measures are designed to allow an organization to measure the total volume of work completed for the CFO over a specific period of time;

- Define projects, players, roles, and responsibilities for financial management systems;

- Ensure resolution of the accounting issues for full implementation of HUDCAPS;

- Ensure proper internal controls are built into the accounting processes and reflected in the financial management systems;

- Ensure proper security controls are implemented, as required by OMB Circular A-130;

- Ensure that the Change Control Board is operating effectively by monitoring approvals for releasing software changes into production and subsequent production problems and failures;

- Require that all information systems for the CFO are placed under Configuration Management software control;

- Implement Help Desk and/or Defect Tracking software to support the HUDCAPS Help Desk and CFO Change Control Board, and build an historical database to assist in resolving production problems; and

- Monitor quality assurance and testing activities.

Recommendations

Acting Director of Information Technologys (IT)

4 (a) Prepare guidelines to assist IT to develop realistic performance standards that are measurable. These standards should address factors in which the contractor can be held more accountable;

4 (b) Incorporate surveillance plans and performance requirements' summaries into performance-based contracts;

4 (c) Develop task specifications that follow internal instructions, IT-CON-04 and IT-CON-05, and include measurable performance standards and acceptance criteria related to each specific task assignment;

4 (d) Ensure that modifications to the task specifications provide additional details about the task and/or work requirements;

4 (e) Ensure that all product acceptance forms are reviewed timely according to agency standards; and

4 (f) Ensure that copies of task specifications and product acceptance forms issued are provided to OPC for their review and evaluation.

Director of the Office of Procurement and Contacts (OPC)

4 (g) Implement performance-based contracting methods for software development and maintenance contracts by:

- Using alternative contract types that will allow the use of deduction schedules and incentives provisions;

- Requiring IT to provide measurable performance standards and acceptable quality levels within the SOW;

- Requiring IT to submit surveillance plans with performance requirements summary within the SOW package; and

- Developing and enforcing oversight controls to ensure Task Specifications are complete and conform to Department procurement requirements.

Auditee Comments and OIG Response

OPC generally agreed with the recommendations of the draft report. IT agreed that the Department should incorporate performance measures, acceptable quality levels, and quality assurance surveillance plans into future contracting efforts. However, performance measures can only be defined in contracts **in cases where Program Area requirements are clearly defined** (emphasis added).

IT's response does not fully address the intent of the recommendations. The intent is not for IT to develop performance standards based on program area requirements. Instead, we are requesting IT to control contractor performance by developing realistic performance standards that are measurable so that contractors can be held accountable for products and services delivered. These factors should include the quality, efficiency, effectiveness, and productivity in providing system support that are independent of program area requirements (see Appendix C on software measurements.)

1. Application system is a set of software designed to support a particular administrative or programmatic function such as accounting, grants management, mortgage insurance, etc.

2. HUD's definition of software development for systems in production would include those activities categorized in FIPS PUB 106 as either adaptive or perfective maintenance.

3. GAO/AIMD-94-115, May 1994, *Improving Mission Performance Through Strategic Information Management and Technology: Learning From Leading Organizations.*

4. Release means moving a set of software changes from the test environment into the production environment.

5. Emergency Release: A release placed into production in less than the normal period of 7 days in advance. This means the required review and testing must be accelerated.

6. Acceptance testing is considered the "last line of defense" for the end user. The end users perform functional tests on the modified software using live data, test data, or a combination of data.

7. Corrective maintenance refers to software changes which are necessitated by actual errors and, therefore, are a reactive process required to keep the system operational.

8. Adaptive maintenance refers to software changes which respond to regulatory or environmental (e.g., hardware, operating system, or data base management system) changes, and which are normally beyond management's control.

9. Perfective maintenance refers to software changes which are executed to meet the evolving and/or expanding needs of the user.

10. Verification and Validation testing is a formal check and balance effort which monitors and evaluates software as it is being built. V&V consists of tasks from a broad spectrum of analysis and test techniques and is performed to determine functionality, uncover performance of unintended functions, and ensure the production of quality software.

11. Library software is a set of programs which organizes and maintains control files of program source-language. Its automated functions include the retention and identification of prior program versions and limited edits over program statement format and content.

12. HUD's definition of software development for systems in production would include those activities categorized in FIPS PUB 106 as either adaptive and perfective maintenance.