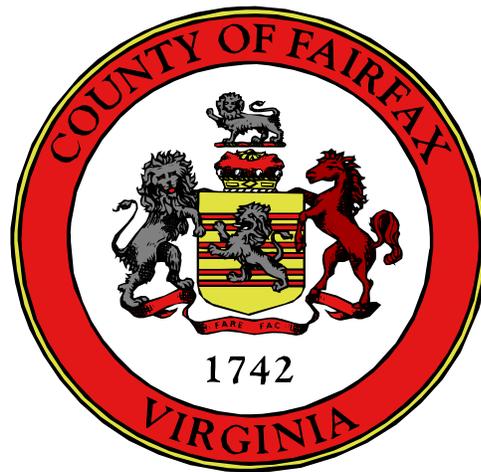


# INTERNAL AUDIT REPORT

## Review of Software Change Management



**FAIRFAX COUNTY, VIRGINIA  
INTERNAL AUDIT OFFICE  
M E M O R A N D U M**

**TO:** Anthony H. Griffin  
County Executive

**DATE:** May 2, 2002

**FROM:** Ronald A. Coen, Director  
Internal Audit Office

**SUBJECT:** Report on the *“Review of Software Change Management”*

Attached is the Internal Audit report entitled, *“Review of Software Change Management”*. It was performed as part of our FY2002 Annual Audit Plan.

The findings and recommendations of this audit were discussed with the Department of Information Technology. We have reached agreement on all of the recommendations and I will follow up periodically until implementation is complete. Their responses are incorporated into the report and the full response is attached at the end of the report. After your review and approval, we will release the report to the Board of Supervisors.

RAC:dbh

## TABLE OF CONTENTS

	<u>PAGE</u>
Introduction	1
Purpose and Scope	2
Methodology	3
Executive Summary	4
Comments and Recommendations	6

## Introduction

Software change is a modification made to a computer program (application). Software changes occur due to legal and regulatory requirements, new products, vendor updates, end user requests, correction of errors, preventative maintenance, etc. The purpose of controls over software changes is to maintain the reliability and integrity of sensitive and mission-critical application systems.

Inadequate control over software change exposes an organization to the corruption of information, which in turn can lead to erroneous management decisions and/or the inability to meet organizational missions. Strong change control processes and procedures are a preventive/detective measure against unauthorized and accidental changes to computer applications that process the County's business. Controls must be in place for software change and the systems that process the County's transactions, just as procedures and monitoring control the transactions themselves.

The importance of software change control cannot be overstated. The Department of Information Technology (DIT) maintains approximately 100 computer application systems, using different languages and accessing multiple databases. These applications run on the following platforms: mainframe, mid-size and personal computers, Local Area Network based client/server, and workstations. The mainframe itself supports the 65 major business and legacy applications and serves over 20,000 agency users at over 200 locations. These systems are the workhorses of the County, running behind the scenes to support the employees in serving the clients and citizens. Some examples of the support functions are: to notify citizens of required real estate payments, cross reference state and federal information for billing and payments, calculate housing subsidies, hold building code and zoning information on land within the County's borders, and process the myriad of federal, state, local and personal payments encompassed within each employee's paycheck.

The Internal Audit Office conducted audits of software change in 1985 and 1989. In addition, the Internal Audit Office led a team in developing the Change Management Policy, Standards and Procedures in 1995. This audit revisited areas of concern identified both in those previous studies and by KPMG Peat Marwick as external auditors for the County's financial statements. The Department of Information Technology is aware of these areas of concern. Its predecessor departments (the Office of Research and Statistics and the Cooperative Computer Center) had attempted several times since 1985 to implement controls for software change. Exhibit A in the report appendix illustrates goals for software change processes and functions according to previous management.

## Purpose and Scope

This audit was performed as part of our FY 2002 Long-Range Audit Plan.

Our overall audit objective was to determine whether adequate controls exist for software changes. Specific objectives included the following:

- To verify that software libraries are controlled and secure
- To determine if modifications are authorized for initial assignment
- To assure that all revised software are tested and approved
- To determine if modifications are authorized for production migration

The scope of the audit was limited to software change control for existing application systems used in production processing. The audit period extended from September 2001 through January 2002.

We researched and identified best practices for software change. These are listed in Exhibit B in the report appendix. The audit included a comparison of DIT policies, processes, and procedures to best practices and an evaluation of DIT compliance with policies, processes, and procedures published in DIT Memo #9, Change Management.

## Methodology

We interviewed appropriate County employees involved with software change. We determined the level of compliance by interviewing the DIT contact listed for a sample of 8 of 72 DIT supported applications. For our sample, we selected applications that run on different platforms and support different agencies. We shared the results of our evaluation with DIT management.

We did not perform the fieldwork phase of our audit due to a lack of key controls in the areas directly related to our objectives. These areas pertain to access restrictions, the Quality Assurance function, and compliance with existing guidance documents. The fieldwork phase is detailed data collection and analysis. It is the process of gathering relevant, and useful evidence and verifying the effectiveness of existing controls.

During the audit, DIT implemented NT Permissions to secure internal programmer and contractor access to the County's servers and the World Wide Web based transaction systems. The Internal Audit Office did not review the adequacy of protection and security as this implementation occurred at the end of the survey phase of this audit.

This audit was performed in accordance with the generally accepted government auditing standards.

Materials used as guidelines for software change management best practices were:

- Federal Information System Controls Audit Manual, January 1999
- Software Change Management: Disaster Recovery Lessons, Gartner Group, October 2001
- Key Practices of the Capability Maturity Model, Version 1.1, Software Engineering Institute, Carnegie Mellon University, February 1993
- COBIT: Control Objectives for Information and Related Technology, Information Systems Audit and Control Foundation, 1998

## Executive Summary

In our opinion, there are few reliable controls in place and operating for modification of the County's application programs. Emphasis must be placed on controlling and monitoring modifications to the systems that process the County's financial and other transactions. The following are identified as areas where controls need to be improved:

- **Quality Assurance** - Large-scale information technology organizations such as the County's DIT are usually supported by a Quality Assurance (QA) function. DIT should institute this function, ensuring a 'sterile environment' for final testing and scheduling of code migration to production and maintaining a consistent process for modifying software. The QA function is a key component of controls protecting the correctness and integrity of source code. The use of a QA function ensures that a group separate from programmers and users migrate code to production and that programmers do not access and modify production data and code libraries through an assortment of processes. DIT should significantly reduce the number of ways that software may be modified so that only an authorized and established method is utilized. Additional resources may be necessary to implement the Quality Assurance function.
- **Access Restrictions** - DIT should implement access restrictions to protect production data and mainframe code libraries. The Remote Access Control Facility (RACF) password system is used to protect the mainframe systems. However, as currently configured, RACF allows programmers open access to production code libraries. In addition, the RACF emergency user passwords are readily accessed by programmers and allow programmers to change or delete production data.
- **Compliance** - DIT management distributed several guidance documents on software change, including DIT Memo #9, Change Management, and the Application Life Cycle Standards. However, the various DIT branches and teams maintaining systems do not consistently follow the procedures in those documents, resulting in an assortment of software change processes. DIT should monitor and assure compliance to software change policies, processes, and procedures. These policies, processes, and procedures should be kept current and reflect best practices for software change, including proper authorization, adequate testing, and timely approval.

DIT management has taken positive steps for software change control by:

- Disseminating the Application Life Cycle Standards, Version 1.1, 2001
- Disseminating the Fairfax County Redesign Work-in-Progress Site, New Upload Process, 2001
- Hiring a new Enterprise Operations Center (data center) manager with in-depth software change experience
- Developing new positions to be used in a Quality Assurance function
- Briefing the entire DIT management team on software change
- Acquiring and implementing Quintus, a tracking system for support calls, and
- Forming a Change Management Committee

Looking forward, implementing a well-designed software change practice should precede any planned migration from the mainframe to other platforms.

## Comments and Recommendations

### 1. **Programmers migrate code into production libraries. There is no separation of duties between software modification and production work.**

Programmers must migrate code into production libraries, as there is no separate group within DIT tasked with code migration. Team leaders determine migration practices and these practices vary by team leader and platform.

Fundamental control standards require that programmers not have direct update capability to production software. This protects the correctness and integrity of the application systems through a division of duties. Updating software or moving additional or revised code into production should be done by an organizational entity separate from and independent of both the users and the programming staff.

The Internal Audit Office performed a review of software change in 1989. In their October 1989 response, the Cooperative Computer Center stated that limiting access to code libraries “would best be accomplished through the creation and staffing of a Quality Assurance (QA) function who would have sole control over these libraries.”

A Quality Assurance function ensures a ‘sterile environment’ for final testing and migration scheduling. The code is no longer accessible by the programmer. Migration schedules and time limits are set up and ‘back out’ procedures are developed to protect the working application system.

Malicious code, the wrong code, or incorrect code could be migrated into production when migration is not performed by a group separate from and independent of programmers and users.

#### **Recommendation**

#### **High Priority**

We recommend that a Quality Assurance function be developed and implemented. The QA function is a key component of controls protecting the correctness and integrity of the source code. The use of a QA function ensures that a group separate from users and development/maintenance programmers migrate code to production. The use of a QA function also ensures that development/maintenance programmers do not access production code libraries.

#### **Department Response**

DIT is in the process of establishing a separate QA function to maintain both QA and production libraries. This would be across all platforms. Developers would migrate code to the QA library for QA testing and final migration to production. A position is being established to perform software change management and migration duties. Due to the number of specialized legacy applications supported, the new procedure will allow flexibility for programmers to have access as required to production libraries in order to remediate an application error and problem with the program on an emergency basis based on approval.

**2. DIT divides access to application code libraries by application groups or teams. However, each of these groups has unrestricted access to their assigned application code libraries on the mainframe.**

Programmers' RACF User ID allows access to development and test environments for their assigned applications. This same user ID allows access to the application's production source code within the mainframe code libraries. That source code defines how the County transacts business, from the payroll to the real estate tax calculations to the balancing of the financial system.

Separate software libraries should be established as a fundamental requirement for maintaining application integrity. These should be separated from each other and from development, testing, and production file storage areas.

Programmers should not have unmonitored and open access to production source code libraries. Access to the separate libraries and file storage areas should be restricted and controlled. Libraries should be protected through a "checked in/out" procedure that maintains the correctness and integrity of the software baseline library.

Financial and confidential County information could be compromised when there are not sufficient access restrictions in place to protect the correctness and integrity of code libraries. Unauthorized or erroneous code could be introduced, affecting the County's financial transactions, or confidential judicial and human services information.

**Recommendation 2a**

**High Priority**

We recommend that DIT further restrict access to production source code libraries so that the integrity of the County's information and processes may be reasonably assured. This requires the separation of duties between software modification, which requires access to development and test environments, and those duties requiring production access.

**Department Response**

Production source code libraries are only accessible by the assigned developers who have specific knowledge of those programs. RACF Security will review all user ID's and access levels. A plan currently under development will be implemented that will restrict access to production libraries.

**Recommendation 2b**

**High Priority**

We recommend that the use of an 'emergency' ID to access application source code should require supervisory intervention, be possible only with the greatest security, expire after a very brief time, and produce an automatic audit trail which is then monitored.

**Department Response**

Emergency ID passwords can only be used by senior programmers for the systems they are authorized to work in and expire after each use. They are required in order to be able to respond appropriately to urgent application support needs, such as program errors and urgent reports required by system proponents. All emergency ID's are audited. A report is generated and distributed weekly. A more rigorous process will be developed to notify managers of emergency ID use. Managers will be required to review the report/notifications. Once the new QA process is in place, use of emergency ID's should be minimal. Adherence to applications standards and procedures is evaluated.

### 3. DIT programmers have unrestricted access to emergency processes allowing them to run production batch jobs or read, alter, or delete production data.

There are two forms of 'emergency user ID' access available to programmers. One form of access called \$Z allows programmers for each mainframe application unmonitored access to alter the jobcards of scheduled production jobs to run those jobs at any time. With this access, programmers are also able to alter the job control language to perform other tasks and then run the job at will. These jobs could read, alter, or delete production data. The chart lists some applications with the number of assigned programmers and programmer accesses in November and December 2001.

System	Programmers Assigned	November 2001	December 2001
AL 2, Personal Property	3	264	284
REA, Real Estate Assessment & Billing	6	235	158
CASPS, County / Schools Procurement	7	82	45
PRISM, Payroll and Personnel	5	76	64
FAMIS, Financial Management	5	46	56

The second form of access called \$E allows programmers for each mainframe application to use their regular RACF user ID to access a password file. The programmer can then use the application \$E User ID with the password to read, alter or delete a data record in the production files. The chart lists some applications with the number of assigned programmers and programmer accesses in November and December 2001.

System	Programmers Assigned	November 2001	December 2001
AL 2, Personal Property	3	21	24
REA, Real Estate Assessment & Billing	6	27	15
CASPS, County / Schools Procurement	7	0	5
PRISM, Payroll & Personnel	5	3	4
FAMIS, Financial Management	5	17	4

RACF access and violations reports have not been produced and used by management since 1998. However, the Information Protection Branch began daily monitoring of the \$E access during January 2002. Both \$E and \$Z accesses were used a total of 1,268 times in November 2001 and 1,077 times during December 2001.

Data integrity depends on specific and limited methods of reading, altering, or deleting data. Although there are valid reasons to have emergency access to data in the production environment, information must be produced and maintained to document and support 'emergency' changes. This information should include the purpose, scope, and authorization of the change and, most importantly, its communication to data owners.

Access to data should be restricted with a thoroughly documented and managed process. This allows management to maintain a complete information trail of data modification that is performed outside the bounds of normal transaction activity.

The integrity of application processing and confidential and financial information is compromised unless emergency access by programmers is restricted.

**Recommendation****High Priority**

We recommend that the use of an ‘emergency’ ID to access production data should require supervisory intervention, be possible only with the greatest security, expire after a very brief time, and produce an automatic audit trail which is then monitored.

**Department Response**

Same as answer in 2b. Emergency ID passwords expire after each use and are audited. A report is generated and distributed weekly. A more rigorous process will be developed to notify managers of Emergency ID use. Conditions that warrant use of Emergency ID’s occur when data may become corrupt or other emergency situations. Managers will be required to review the report/notifications.

**4. Some of the application programming teams do not consistently follow software change policies and procedures. In addition, DIT software change policies and procedures do not, at present, reflect key aspects of best practices for software change.**

Various DIT branches and teams do not consistently comply with software change policies and procedures.

- Not all teams follow emergency change or user acceptance procedures as documented in DIT Memo #9, Change Management.
- Not all team leaders monitor the writing of changed code or the movement of changed code into production.
- Not all team leaders and programmers list or describe a requested change on the change management form as required by DIT Memo #9, Change Management, or by using Quintus as recommended in the Application Life Cycle Standards.

DIT policies, processes, and procedures, as documented in DIT Memo #9, Change Management, and the Application Life Cycle Standards, do not reflect key aspects of best practices for software change management. The following are not included:

- Prioritized Change Requests
- Programmer Test Plans
- Quality Assurance function

Monitoring compliance with policies and procedures is a control and provides a basis for continuous improvement. Effective review by management helps to prevent or detect unauthorized or erroneous actions. It ensures that management is aware of business practices and can determine actions necessary to correct non-compliance with policies and procedures.

Policies and procedures help provide assurance that adequately designed and written code is used to update application programs. Policies and procedures, over time, may no longer be followed when compliance is not monitored.

**Recommendation****Medium Priority**

We recommend that management monitor programming staff and contractor work practice to assure compliance with software change management policies, processes, and procedures. These policies, processes, and procedures should reflect best practices for software change, including proper authorization, adequate testing, and timely approval.

**Department Response**

Currently, the business and technical managers approve the change request and staff enter their approval into the change management system. Before change is migrated to production, the business user must sign the request to provide approval of the testing and consent to migrate the changes to production. These practices will be reviewed for consistency with the revamped change management policy. The new procedure will be distributed to and reviewed with staff. Accountability for application development standards including procedures is reflected in performance evaluations.

**Exhibit A**

<b>Essential Functions</b>
Access control by application group and product interface with RACF.
Established migration path from test to production and return for all software types.
Audit trail down to lines of code level. Non-programmer dependent.
Software unit interrelationships to provide impact analysis for modifications.
Control for all software units – source, load, proc, parm, copy, and database entities such as the Data Dictionary.
Automated project leader/supervisory authorization for movement to production.
Direct relationship between source and load established such that it can be guaranteed that the two are in synchronization. This must be an inviolate and auditable link.
Ability to track change activity by software units and users.
Product must interface with CA-Librarian to facilitate conversion.
CA-IDMS/R component support. Changes must be tracked as they happen and not ‘after the fact’.
DB2 component support.
Interface to Info/Management (Problem and Change tracking package used by the CCC).
Automated software management for the PC/LAN and workstation environment for interfacing with the mainframe.
A consistent interface across all facilities of the automated software package (including ISPF, IDMS, and DB2).

**Exhibit B**

<b>Minimum Requirements For a Best Practices Process Model</b>	
1.	Documented request from user agency or determination from DIT.
2.	Users prioritize change/enhancement with standard and consistent methodology.
3.	Programmers develop change/enhancement with supervisory review.
4.	Programmers test enhancement with a test plan and with supervisory review.
5.	Migration to acceptance test environment with supervisory review.
6.	Users test enhancement with a test plan.
7.	Users accept and approve enhancement in writing.
8.	Quality Assurance Review by group separate from programmers and users.
9.	Migration to production by group separate from programmers and users.
10.	Update documentation and training to reflect change.
11.	Change request is closed.

In addition, these controls should be included:

- Restrictions to control library access
- Separation between libraries, such as development, test, and production, so that production is updated by an assigned group separate from programmers and users