



Fairfax County Internal Audit Office

Department of Information Technology
Software Change Control Audit - Mainframe Systems
Final Report

March 2007

"promoting efficient & effective local government"

Introduction

Software change involves modifications made to computer application programs. These changes occur due to legal and regulatory requirements, new products, vendor updates, end user requests, correction of errors, preventative maintenance, etc. Effective controls over software changes are needed to ensure the reliability and integrity of sensitive and mission-critical application systems.

Inadequate control over software change exposes an organization to potential 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. Without these controls, it is possible that unauthorized changes could be made to produce payments that go undetected.

Software change controls are critical to the county's business processes. The Department of Information Technology (DIT) maintains approximately 400 computer application systems, using different languages and accessing multiple databases. These applications run on the following platforms: mainframe, mid-range, network based client/server, and workstations. The mainframe itself supports approximately 80 major business and legacy applications and serves over 20,000 agency users at over 200 locations.

Executive Summary

We last performed an audit of application software change controls in fiscal year 2002 and noted at that time that there were few reliable controls in place. While DIT has made progress in this area, we found that sufficient and reliable controls were still not in place and operating to ensure the propriety of mainframe software change control processes. Emphasis should 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:

- **Separation of Duties** - Information technology organizations such as the county's DIT are usually supported by a Quality Assurance (QA) function. 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. While DIT has made progress in the implementation of the QA function, programmers were still relied upon on to make the code changes and migrate code into production. Additional resources may be necessary to fully implement the Quality Assurance function.
- **Audit Trail** - A log was generated every time an emergency user ID (EMGID) was used and the report was sent to the branch manager on a daily basis for justification. An audit trail should specify when the event occurred, who the user was, and a

record of the actual change made to the code or the data. However, the EMGID log only stated who used the EMGID and when, it did not record what had been done to the system.

- **Compliance** - The various DIT branches and teams maintaining systems did not consistently follow established procedures for performing software change processes. This resulted in a variety of operating processes among the different groups. DIT should monitor and assure compliance with 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 controlled migration to production.

While our efforts were focused on the mainframe computing environment, weaknesses in the software change controls for client server and web based applications should also be addressed by DIT.

Scope and Objectives

This audit was performed as part of our fiscal year 2006 Annual Audit Plan and was conducted in accordance with generally accepted government auditing standards. The audit covered the period of January 2006 through May 2006, and our audit objectives were to determine that:

- Prior recommendations from the FY 2002 Review of Software Change Management have been implemented
- Authorizations for software modifications are documented and maintained
- All revised software are tested and approved
- Software libraries are controlled and secure

Methodology

As a preliminary step we examined the implementation status of the Internal Audit Office's (IAO) recommendations to the DIT as part of the Review of Software Change Management that was completed on May 2, 2002. We reviewed the change management policies and procedures, and interviewed appropriate employees to understand software change management process. We determined the level of compliance by interviewing appropriate county employees involved with mainframe software change, i.e. quality assurance coordinator, security administrators and mainframe applications developers. 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 separation of duties, emergency User ID usage, audit trail, and compliance with existing guidance documents.

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 2006
- Institute of Internal Auditors: Global Technology Auditing Guide 2 – Change and Patch Management Controls 2005

The Fairfax County Internal Audit Office is free from organizational impairments to independence in our reporting as defined by Government Auditing Standards. We report directly and are accountable to the county executive. Organizationally, we are outside the staff or line management function of the units that we audit. We report the results of our audits the county executive and the Board of Supervisors, and reports are available to the public.

Findings, Recommendations, and Management Response

1. Separation of Duties

While there were two full-time staff members assigned to the Quality Assurance (QA) function, QA did not have sufficient control over the code migration process. The same programmer who made the changes to the application also migrated and deployed the code into the production, resulting in a lack of separation of duties. The practice in place was for change requests to be discussed at weekly change management meetings where information, concerns, and comments were shared in order to eliminate potential disruptions of service to the county IT environment. Changes were then being made by programmers to the production environment after they were approved at those meetings.

Fundamental control standards for application system integrity require that programmers not have direct update capability to production software. 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. Best practices for a Quality Assurance function dictates that there be a secure 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.

In 2004, during a follow-up review to our prior software change control audit, we noted that DIT was in the process of implementing a pilot project using the IBM Software Configuration and Library Management (SCLM) tool for the QA function to manage version control, track changes and perform migration. However, the SCLM project was put on hold and not scheduled for resumption until January 2007.

We also noted that the programmers were using emergency user IDs to move planned changes from development or acceptance environment to production environment. It is not uncommon for program changes to be needed on an emergency basis to keep a system operating. DIT Change Management Policy/Procedure states that an emergency change is a change to the IT environment or infrastructure that cannot wait

until the weekly change management meeting.

Recommendation: We recommend that DIT use the QA function to perform final testing and migration of code to production for planned software changes. The QA function should be a key component of controls protecting the correctness and integrity of the source code by having a group separate from users and development/maintenance programmers migrate code to production. The QA function should also ensure that development/maintenance programmers do not access production code libraries.

We recommend that DIT prohibit the practice of using emergency user IDs for routine, planned changes.

Management Response: DIT agrees that best practices are to have an independent QA function that performs migrations of programming code changes from either development or acceptance regions to the production regions. After the 2002 audit, DIT intended to establish and staff a Change Control function, and to implement the IBM SCLM (Software Configuration and Library Management) utility. However, as a result of significant budget reduction in FY03 and FY04, DIT's flexibility for properly establishing and staffing this function was reduced. In lieu of a fixed function, DIT has established a Change Control team which meets weekly to disclose and schedule all upcoming migrations to production. However, the SCLM tool was not implemented due to conflicts with other project priorities, cost versus benefit considerations based on continuing strategy to move applications off the mainframe environment, and the lack of dedicated staffing. As we continued to look at this, we have noted that there is no known history of negative incidents associated with the lack of a dedicated function for either bad code migration or fraudulent activity. We contend that without a separate function, that process between the DIT programmers and agencies supported for mainframe based applications required a tightly controlled and efficient process with confidence and detail knowledge needed to accomplish migrations within the application programmer groups.

DIT does employ separation of duties and migration controls for many systems and environments. This includes the use of PVCS and separate staff for migrations of web-based applications, and the existing QA function does separately migrate changes to WebMethods application to production. However, the QA function has not been fully staffed and trained to independently perform mainframe migrations.

Fairfax County is experiencing a future trend that will result in fewer mainframe platform applications. It may not be a worthwhile investment of resources to significantly enhance migrations for the mainframe platform. In FY08, DIT will explore ways to further improve the separation of duties for FAMIS, CASPS and other mainframe code migrations. We will continue to look at several options to minimally address the core issue of the audit finding:

- Staff position reallocation availability in establishing an independent QA and migration function to be managed either within TID's production operations area, from a new function are outside BSD or ESD; or
- Using a peer approach which would designate a group of certain

programmer/analyst to migrate production changes made by programmer/analyst in another Branch or group not associated with the code development.

The anticipated completion date is December 2007.

2. Emergency User ID Audit Trail

A log was generated every time an emergency user ID was used and the report was sent daily to the branch manager of the employee for justification. However, the EMGID log only stated who used the EMGID and when, it did not record what had been done to the system. The application programmer who used the EMGID was responsible for providing an explanation of updates made to the system with no independent verification. Without a sufficient audit trail and timely, independent review, unauthorized or erroneous code could be introduced, affecting the county's financial transactions, or confidential judicial and human services information.

Application developers must use the application's EMGID to access application source code libraries and production environment. This can be the result of:

- A group of customers or a critical customer is completely out of service.
- Malfunction with hardware.
- Severe degradation of service needing immediate action.
- A system/application/component is inoperable and the failure causes a negative impact.
- A response to a natural disaster, or
- A response to an emergency business need.

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.

Audit trails can provide a means to help accomplish several security-related objectives, including individual accountability, reconstruction of events, intrusion detection, and problem analysis. An audit trail should include sufficient information to establish what events occurred and who (or what) caused them. In general, an audit trail should specify when the event occurred, the user ID associated with the event, the program or command used to initiate the event, and the result.

Recommendation: We recommend that emergency user ID audit trails not only record who uses the EMGID and when, but also record what has been done to the system. The audit trail should be reviewed and analyzed in a timely manner by someone other than the programmer who uses the EMGID.

Management Response: Use of the Emergency ID was established to do two things. Regular IDs for programmer/analysts do not have the authority to migrate programming changes to production so they are forced to use a new ID and create a special event.

And use of the Emergency ID forced the generation of an audit log for management attention and review. Thus, the 'emergency ID' is used for routine code migration process to give a virtual segregation of duty between development and migration tasks. If an independent mainframe QA function was established with dedicated staffing and a migration control utility, programmer/analysts would no longer need to use the emergency ID for over 90% of the migrations to production (the emergency ID would then be reserved for true emergencies that could not wait until at least the next business day).

The current Audit Report does use SMS data and reports who used the Emergency ID and when it was used. DIT will investigate enhancing either the report or the manual process to also capture a record of the loadlibs and modules that were affected and other reasons for and comments about the change. Ideally, a migration control utility like SCLM would be used to automatically record the additional information; however, because of the unique system architectures, even SCLM could not automatically record all changes to CASPS.

DIT will also investigate taking steps to reduce the reliance on the Emergency ID. If the earlier changes to improve the separation of duties are achieved, there will be fewer incidents of the emergency user ID being activated. The anticipated completion date is October 2007.

3. Compliance With Procedures And Best Practices

We noted that the level of compliance with established change management procedures varied by the branches and application developer groups. DIT management had instituted two procedures, Memo #9 - Change Management and Memo CM001 - Change Management Policy/Procedure, that defined requirements and responsibilities for monitoring the work of staff and contractors to ensure compliance with the change management policy/procedures. As part of these procedures, branch managers for DIT support groups received reports of activities that impacted the change management process for applications in the mainframe environment. Branch managers were expected to respond to the Quality Assurance Office to explain the reasons for such activities. However, we noted that not all change requests had the corresponding change request forms, documented test plan, and user sign-off before the changes were made to the production environment.

In addition, IAO researched and identified a list of best practices for software change management. These best practices were listed in Exhibit A alongside current DIT policies and procedures. The comparison showed numerous instances where the DIT policy and procedures could be strengthened.

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 and provides assurance that adequately designed and written code is used to update application programs. It ensures that management is aware of business practices and can determine actions necessary to correct non-compliance with policies and procedures.

Recommendation: We recommend that management monitor programming staff and contractor work practices 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, timely approval, and the use of a QA function to perform final testing and migration of code to production.

Management Response: The FAMIS & CASPS application teams are required to follow policies and procedures for software change and migrations. The FAMIS team, in an effort to move toward less paper, was obtaining some user approvals via e-mail, and some of these were not filed in hard copy. The FAMIS team has changed their procedure to be in compliance.

In FY08, DIT will review and modify the Change Management Memo #9, and the Change Management Policy/Procedure Memo CM001 to enhance and/or make them more consistent. If possible, they will be consolidated. These changes will be reviewed in draft with Internal Audit before they are re-published. The anticipated completion date is July 2007.

EXHIBIT A

	Minimum Requirements For a Best Practices Process Model	DIT Memo #9 Change Management	DIT Memo CM001 Change Management Policy/Procedure
1.	Documented request from user agency or determination from DIT.	Documented.	Documented.
2.	Users prioritize change/enhancement with standard and consistent methodology.	Documented.	Documented.
3.	Programmers develop change/enhancement with supervisory review.	<i>No mention of “supervisory review.”</i>	Documented.
4.	Programmers test enhancement with a test plan and with supervisory review.	<i>No mention of the “test plan.”</i>	Documented.
5.	Migration to acceptance test environment with supervisory review.	<i>No mention.</i>	Documented.
6.	Users test enhancement with a test plan.	<i>No mention of the “test plan.”</i>	<i>No mention.</i>
7.	Users accept and approve enhancement in writing.	Documented.	<i>No mention.</i>
8.	Quality assurance review by group separate from programmers and users.	<i>No mention.</i>	<i>No mention.</i>
9.	Migration to production by group separate from programmers and users.	<i>No mention.</i>	<i>No mention.</i>
10.	Audit trail tracks all the software changes down to lines of code level.	<i>No mention.</i>	<i>No mention.</i>
11.	Update documentation and training to reflect change.	<i>No mention of the “training.”</i>	<i>No mention.</i>
12.	Change request is closed.	<i>No mention.</i>	Documented.