The Fine Art of Testing Software Applications

Overview
Whether software is acquired or developed, it needs to be tested to ensure it meets the needs of your organization. Without adequate testing, software can produce erroneous results or even damage your image and reputation.

Software testing is an investigation that provides information about the quality of the application. This information allows business managers to understand the risks associated with the application.

Software Controls
Organizations implement software controls to manage and reduce their risks. These safeguards are classified as preventive, detective, or corrective controls.

Preventive controls prevent errors and irregularities and are the most preferred. They also tend to be the least costly to implement. An example of a preventive control is validation that occurs at the time of data input where the data entered is validated to ensure that only numbers are entered in numeric fields.

Detective controls identify errors and irregularities. An example of a detective control is a batch process that counts the number of transactions and creates a total of all dollar amounts. This figure is then then compared to a pre-determined total to ensure that all transactions were received and processed.

Corrective controls include processes and procedures to correct issues and resume operations. An example of a corrective control is a routine that identifies an out of balance condition and stop processing until manual intervention corrects the situation. Corrective controls tend to be the most expensive.

Many software applications include all three of the above controls. A robust software testing process will review the controls to ensure they are appropriate for the environment and sufficient to meet management's objectives.

Testing Opportunities
Testing can take place at various stages during the software development lifecycle:

- System design – software controls should be identified at the time the system is designed. Before the software has been written, a manual process can review the planned controls. This ensures that controls are built into the system as it is developed.
- Programming – when a programmer tests an application as it is being developed, it is called program testing or unit testing.

- Integration testing – testing the interfaces between programs is called integration testing and verifies the interfaces between components.
- System testing – system testing evaluates a completely integrated system to verify it meets requirements. Once the programming or acquisition is complete, testing is performed on the entire system, ensuring that all controls operate as desired and errors are handled gracefully, without abnormal program termination.

Testing Responsibilities

The responsibility to test software applications falls on many different individuals. It is important that testing be performed by not only by the programmer or developer, but also by other entities within the organization. Larger organizations may have a separate testing or quality assurance department while smaller businesses will only have one or two additional staff members that assist with the process.

Testing can never completely identify all software defects. It provides information on the state of the application which might indicate that there is a problem or issue that needs to be addressed.

Testing cannot confirm that software functions properly under all conditions. Instead, it identifies situations where the application does not function properly under specific conditions.

Testing Methodologies

There are a number of methods available for testing software applications. The specific steps chosen depend on a number of factors including the project size, anticipated project life, resource availability, budget, and other factors.

Depending upon the software development models used, testing will occur at different stages of the software development process. In a traditional software development model, a majority of the testing occurs after the system has been coded. In newer development models, the programmer performs a majority of the testing as the code is developed.

To ensure the best results possible, testing should be a planned process and include:

- Scope - identify the type of test to be performed, applications involved, timeframe, resources required, and testing requirements.
- Preparation – assemble or prepare information needed for the test including data input forms, totals, credentials (IDs and passwords), prepare test cases, decide if automated testing software should be used, prepare systems for testing, write test scripts, back up data, etc.
- Testing – perform testing and compare with expected outcomes.

- Report – analyze test results and prepare a report of findings and recommendations.

Functional Testing
The System Design Specification and/or programming documentation often contain information on the functionality of the application. Testing that verifies a function of the code is considered functional testing. Functional testing answers questions such as:

- Does the input form ensure only numbers are entered in numeric fields?
- Do error routines work as planned?

When performing functional testing, both valid and invalid data should be entered in all appropriate fields. By entering both valid and invalid data, the tester can:

- Identify normal operation of the software application
- Verify input validation routines work as desired
- Assess error message and error handling processes

Non-functional Testing
Software scalability and performance evaluations are considered non-functional testing. Non-functional testing refers to testing that is not related to the use of the application. For example, it evaluates software performance under loads, user experience, security testing to protect against intrusion attempts, and if the application performs well under both under usual and unusual conditions. Non-functional testing can refer to both the software itself as well as the related infrastructure. Non-functional testing answers questions such as:

- As the number of simultaneous users increases, at what point do response times exceed 5 seconds?
- Are the bottlenecks related to Internet connectivity, database access, or inefficient software code?

Error Messages
Error messages need to provide enough information to a legitimate user that they can correct their mistakes. However, for security reasons, the error messages and error handling may need to be more generic. For example, if a user was not able to log on successfully the message should read something similar to "Unsuccessful logon: Invalid User ID or Password." This type of message provides information to legitimate users while also limiting the amount of information provided to hackers. With this type of error message, the hacker doesn't know if it is the ID or Password that is invalid, thus increasing the effort required to break into the system.

The software application's error handling routines should gracefully handle all expected errors. For example, if a user accessing a web site clicked on a broken

website link (hyperlink), the user should not receive a system generated 404 error page. The same holds true for all types of common errors including 503 Service unavailable/Forbidden. Instead of relying on the system to present an error message, the application should gracefully handle the error by directing the user to a custom page.

Summary

Software applications have unique custom code that is vulnerable to a variety of risks including functionality, security, and performance issues. A planned approach to software testing can help manage these types of risks. While testing can never completely identify all software defects, it can provide the organization with information on the state of the application and indicate if there is a problem or issue that needs to be addressed. Web application security audits help identify risks before they can be exploited, thus helping protect the organization's image and reputation.

Publication Information

Altius IT is a security audit, security consulting, and risk management firm. We are certified by the Information Systems Audit and Control Association (ISACA) as a Certified Information Systems Auditor (CISA), Certified in Risk and Information Systems Controls (CRISC), and Certified in the Governance of Enterprise Wide IT (CGEIT). For more information, please visit www.AltiusIT.com.