How to Develop Powerful Test Cases

By Joe Zec  Feb 24, 2014 2:44 pm PST

Peer Reviewed: Software Validation

In today’s life sciences industry, software is playing an increasingly critical role in business operations. From clinical trial data collection to complaint handling systems, from laboratory information systems to manufacturing execution systems, software is growing in size and complexity, presenting a validation challenge. Validation frameworks appropriate for small, straightforward software systems may not be suited to today’s enterprise systems.

The installation qualification (IQ)/operation qualification (OQ)/performance qualification (PQ) validation framework has its roots in process validation in the pharmaceutical industry (1). Although the 1987 FDA Process Validation Guidance does not mention operational qualification, both installation qualification and performance qualification are defined in terms of process capabilities. It was a natural evolution to extend this framework to validation of process control systems, where software often takes the form of firmware controlling manufacturing equipment or other mechanical systems. However, the extension of IQ/OQ/PQ to the testing of large information technology (IT) systems has turned out to be a poor fit.

IQ/OQ/PQ became institutionalized thanks to the GAMP Guide for Validation of Automated Systems (2). Manufacturing and process personnel who migrated to IT brought this testing framework with them as the only methodology they knew, yet the enterprise-wide systems they worked on demanded a higher level of testing thoroughness than IQ/OQ/PQ provided. Thorough testing is key to the successful deployment of a complex software system, so it would be appropriate to apply a testing framework that encourages it.

Requirements coverage is a testing framework that encourages thoroughness. In this methodology, test procedures are developed such that each and every software requirement is addressed by at least one test. Not only does this promote the development of a thorough suite of test procedures, but requirement-based test development has the added benefit of testing the system against how it was supposed to be built, not how it was actually built. Requirements coverage is often measured by way of a trace matrix wherein links are created between requirements and tests, which thus allow for requirements without tests to be identified and rectified.

Requirements coverage assures that each requirement has a test, but it does not assure that each requirement has sufficient tests. Test type coverage is a method that can be combined with requirements coverage to address this shortcoming. In this technique, a software requirement is analyzed to determine what test type apply to it. There are many different types of tests that can be applied to software. Some of the more common types include functional, error, stress, negative, volume, boundary, performance, and operational...
tests. By reviewing a requirement and its associated tests against a comprehensive list of test types, one can become assured that all of the test types applicable to that requirement have been covered.

These two coverage frameworks are well suited to black-box testing of a system, where access to or even knowledge of the source code is unavailable. In white-box testing, however, structural coverage is a superior framework in which to achieve thorough testing. In this method, tests are developed that force execution of source code statements to varying levels of rigor. Some of these levels include statement coverage, decision coverage, and condition coverage (3). Using a software tool to instrument the software being tested and then executing the test procedure suite is often how structural coverage is measured. The instrumentation tool will then identify lines of source code or other structural characteristics not executed by the test suite.

Whether you are doing black or white-box testing, the principle is the same: utilize a testing framework that fosters thoroughness in testing. This thoroughness is a critical element in determining if a complex software system is fit for its intended use.

References

1. FDA, Guideline on General Principles of Process Validation (Rockville, MD, May 1987).
3. FDA, General Principles of Software Validation; Final Guidance for Industry and FDA Staff (Rockville, MD, 1992).

Facebook Twitter Pinterest Email Add This

Tags: Computer & Software Validation, GXP

Joe Zec
Joe is currently senior manager of Validation with Philips Healthcare’s Quality and Regulatory department. He and his team are responsible for the validation of software systems used to automate various functions of...

View Author Bio

Source URL: http://www.ivtnetwork.com/article/how-develop-powerful-test-cases