November/December 2010 $9.95 www.StickyMinds.com

BETTER SOFTWARE

The Print Companion to STICKYMINDS.COM™

TESTING GONE WILD
Regulated software unleashed

THE OPTIMIST'S DILEMMA
Looking on the (not-so-bright) side

ISTOCKPHOTO

"We can't do that, we'll never pass the audit!" If you've worked in regulated software, chances are you've heard that statement. The threat of an audit can cause even the most enthusiastic proponents of new ideas to cower in fear of the dreaded "finding," an auditor's report on something noncompliant in your process. The truth is, many of the most cited "can'ts" about regulated software testing are nothing more than myths.

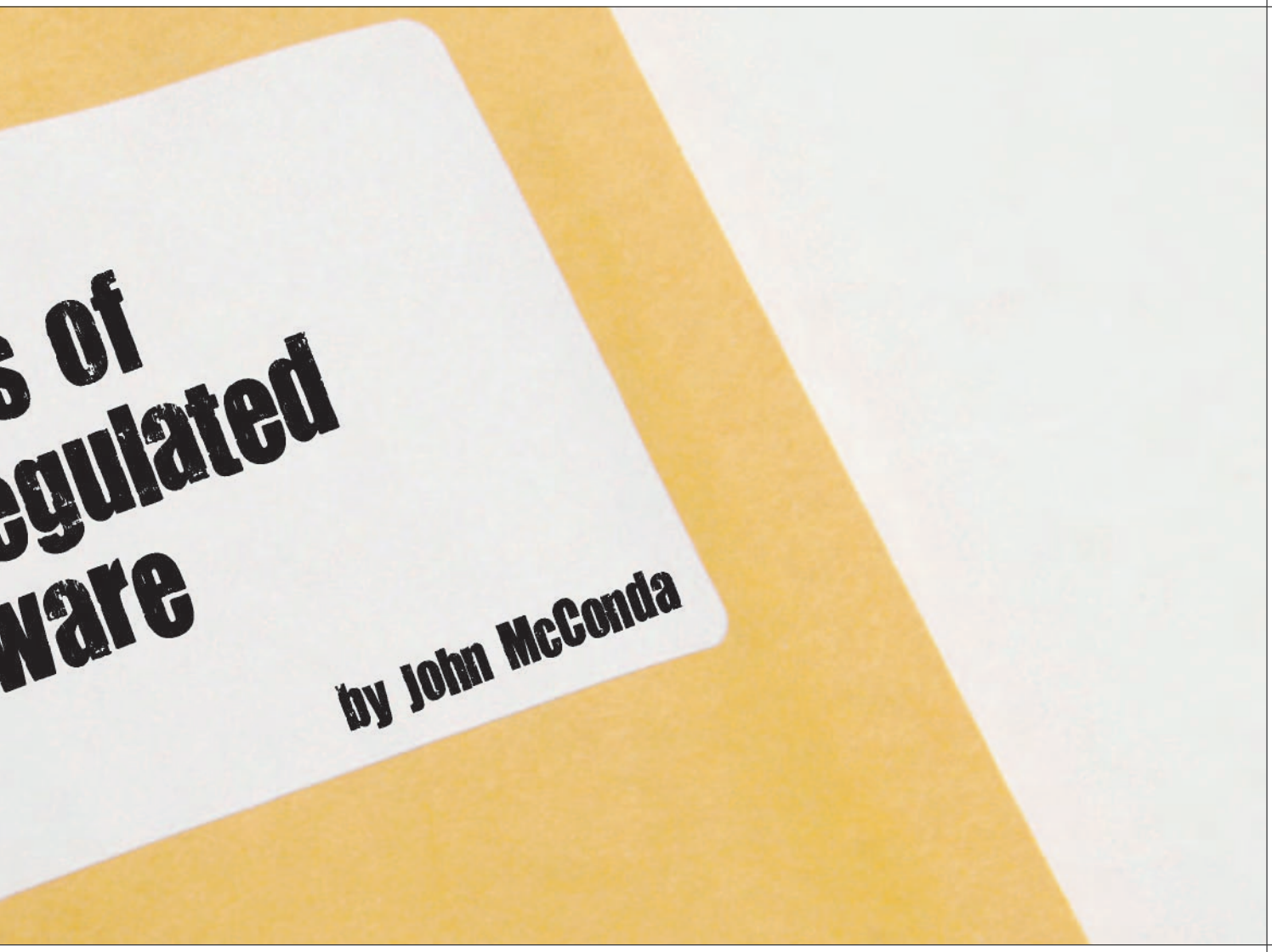
Myth 1: We Can Only Test with Prewritten Test Cases

Scripted manual tests have been used for so long that many organizations assume that they are the only option, but that doesn't mean this is the only acceptable approach. When someone says, "We can't do exploratory testing," what he is usually afraid of is not being able to present objective evidence of testing, such as the kind called for in FDA regulatory guidelines [1]. Those regulations require that someone looking at your documentation can tell what you tested without taking your word for it. There is nothing about prescripted tests that make them better for this than exploratory tests. Some problems with test evidence, like putting a checkmark in the Pass box or just typing "as expected" for a result, are actually unique to prescripted test cases. In this regard, exploratory testing can be more suited to showing objective evidence because notes on observation make up the bulk of the documentation. An exploratory test session is designed so that the tester has objective evidence about what he saw.

Screen and video capture tools can make gathering objective evidence even easier. One tool, SiriusQA's Test Explorer, is specifically designed to support the recording of objective evidence during exploratory testing [2]. Other tools I've used that feature video capture are BlueBerry Test Assistant and HP Quality Center.

Myth 2: Test Automation Is Too Difficult

Test automation is a tricky road to walk in any context, but in the regulated world, misunderstood rules often remove it from consideration completely. This is unfortunate because automation can supplement a regulated testing process just as much or more than a nonregulated one. An automated test can add control to your evidence because it logs results the same way every time, but some auditors may ask you to validate each tool for your specific process to make sure it meets their standard of objective evidence. Types of tool validation that I have seen are *general* and *specific*. General validation is usually a certification for a certain industry. Some companies will tout this certification about a particular tool if it has passed an audit by a regulatory body in that industry. Second, the tool may need to be validated for your specific



environment. In my experience, our team had to prove that each automated test's results were identical to the same test run manually. If your regulations require a high level of validation before use, you may have a difficult choice between the amount of time automation will save you in the long run against the additional cost you will incur for tool and test validation. On the other hand, if you are spending a substantial amount of time running a large suite of regression tests manually, you may still come out ahead.

Automation is more than just regression testing, though. It can also be used to run high volumes of test data through thousands of combinations that would take months to do manually. In his article for StickyMinds.com, "The ROI of Test Automation," [3] Mike Kelly lists even more types of test automation, many of which, if done in addition to a validated manual test process, may not even need to be a part of your officially validated process (see Myth 5).

Myth 3: Regulated and Agile Don't Mix

Over the past ten years, agile software development has taken the world by storm, but regulated software has been one of the last holdouts. Some of this can be attributed to one of the Agile Manifesto's stated core values of "Working software over comprehensive documentation [4]." As agile processes have evolved, so have ideas about where they can be used. Today, FDA-regulated companies like Abbott Labs are presenting case studies that describe how agile processes have been used in developing their applications [5], and Department of Defense-regulated companies like Lockheed Martin are also publishing case studies of their results with agile [6]. The common thread among many of these case studies is that software teams were able to cut through perceived notions and get to the real purposes and constraints of the regulation. Once those were understood, then it was possible to tailor agile processes to the regulatory requirements.

Rich Sheridan, CEO of Menlo Innovations, a biotech software firm, believes agile processes are essential to software that is developed for medical devices. He talks about many agile ideas his teams implement like "weekly show and tells where the customer runs the software in front of our team," automated unit testing, and paired programming. He adds: "In our environment, no production code can be produced by anything other than a pair of developers, thus each line of code is reviewed by two sets of eyes. Regardless of what the FDA may think about such a system, I know I wouldn't want to have a device hooked to me that wasn't developed in a test-driven, paired-programming environment [7]."

Myth 4: The Auditor Is Your Enemy

When most of us hear the word auditor, we think of a

grumpy IRS agent with horn-rimmed glasses and a thin black tie. In my experience, the auditors I've encountered were quite willing to work with our team to make the process as smooth as possible. The most effective way to avoid significant findings is to know what you're being judged on as you conduct your process and to make sure everyone on the team does, too. At one of my former companies, we put together an agenda for the auditor to approve and then put all of our evidence into binders in the conference room so she could peruse at her convenience. We made every effort to be transparent, so she had no reason to suspect we were covering up a flaw in our process. If we did have a known flaw, we would document it for corrective action up front. Practitioner and author Matt Heusser, who has tested systems subject to HIPPA and SAS 70 audits, believes the perception of an adversarial relationship with auditors is counterproductive. "There is the perception that you never volunteer information to an auditor, so we were taught to answer questions and say nothing more. There was even a joke that, if an auditor asks if you know what time it is, the correct answer is "Yes." But even the word "audit" itself, comes from the Latin word for hearing. An audit should be a software team's invitation to "Come hear what we do [8]."

Myth 5: Regulated Testing Is Boring

A few years ago, when I was working on my master's degree, I talked after class one evening to a few fellow students who mentioned they were testers. Excited about having classmates in the same field as mine, I asked them how they liked their jobs. All three of them agreed, "I'm getting this degree so I can get out of testing. It's so boring!"

Bored often means you're not using your brain. Regulated testing is subjected to this stigma even more often, mostly due to Myth 1. Doing nothing but executing the same manual scripts over and over is enough to make any tester consider taking apart his cubicle with a screwdriver. Despite the ideas I described in Myth 1, your company may not be ready to move away from test scripts for their officially validated testing. If that's your situation, then there's always what I like to call Black Ops testing. The first time I brought up the idea of using exploratory testing for our regulated software, I encountered some resistance to making it part of our standard operating procedures. What we agreed to was a clause in our documentation stating that we "reserve the right to do other forms of testing as needed." This gave our testing team carte blanche to do as much additional testing as we needed to feel good about our coverage, without the burden of test cases. To succeed at this, though, you need to have time for your extra testing built into the project plan. If your project manager balks at this, make a case for just a few extra hours the first time. If you can show how much more efficient your testing is becoming, you'll have leverage to get the time you need.

Certain software is regulated for a reason. These applications are often life critical or they process sensitive information. Stakeholders deserve the best testing possible, performed by the best testers around. Unfortunately, the myths around regulations can keep good testers away and handcuff those who do test them. Regulations are intended to provide accountability, but it is up to those of us who create and maintain these applications to make ourselves accountable for testing with the best tools and techniques available—and not to be intimidated by myths.

{end}

jmcconda@gmail.com

Sticky Notes

For more on the following topic go to www.StickyMinds.com/bettersoftware.

- References