eGUIDE

Increasing Code Quality with Automation and Machine Learning

Launchable

In the struggle to ship code out the door faster and faster, are you also creating a potential future time bomb ignoring code quality? Sometimes it can be challenging to get buy-in about the value of high-quality code. Many short-sighted managers are only worried about the here and now - and don't see the growing snowball of technical debt that is accumulating and will eventually come rolling their way. This eGuide has articles, case studies, and other resources that speak to the value of code quality and how you can use automation and machine learning to optimize it.

Increasing Code Quality with Automation and Machine Learning

6 Steps for Succeeding with Test Automation in Agile

Lots of test automation efforts in agile software development fail, or at least do not maximize their potential. This article looks at two main reasons test automation may not live up to the expectations that testers and other stakeholders in the agile development process have, then outlines six steps to avoid falling into these traps. Here's how to succeed with test automation in an agile environment.

How to Build a Data-Driven DevOps Decision Making Culture

The growth in AI and machine learning solutions can enable teams to create a modern data-driven DevOps culture that streamlines development processes significantly, maximizes resources, and improves software quality.

Dealing with a Test Automation Bottleneck

The test team uses the test automation system to execute thousands of test cases because ... why not? The tests are running automatically, for free, so there is no incentive to improve test efficiency. Just run them all! But eventually, as more and more tests are added, the system becomes overloaded. Test runs are delayed and you get a bottleneck. Don't throw more money—or new systems—at the problem; do this instead.

The Evolution of DevOps Testing Tools

The DevOps ecosystem has seen two evolutions of tools, and we are on the cusp on the third evolution. Within these three evolutions of DevOps tools there are established tools in the various channel types, from coding and code analysis, to build and QA, and overall operations.

How AI is Transforming Software Testing

The time needed to complete the slew of required test cases directly conflicts with the fast pace driven by agile-like frameworks and continuous development. The exploration of alternative and superior testing methods, such as automation and AI, is now a necessity in order to keep pace and equip QA and test teams with augmented efficiencies.

How to Deliver Code Faster With Test Automation Powered by Machine Learning

How does predictive test selection help prioritize the most important tests to run, ultimately saving you time and resources while delivering software faster?

What's Our Job When the Machines Do Testing?

While many of the tasks across our diverse test practices are similar, each of our jobs has unique challenges, so priority will be determined by the specifics of our organizational context. So your job, when the machines can do testing, is to figure out what tasks you want them to do for you.

Defensive Design Strategies to Prevent Flaky Tests

More testing isn't always better testing. If your tests are unreliable, they can cause more harm than good. "Flaky tests" aren't as rare as we'd like to think, and there are a few things you can keep in mind to avoid them and help your development lifecycle move more smoothly.

6 Steps for Succeeding with Test Automation in Agile

Bas Dijkstra

In order to keep up with the ever-shorter release cycles that come with the adoption of agile software development, many development teams are embracing test automation as a means to continuously ensure that every software release conforms to the desired level of quality.

This is a significant shift from traditional software development practices, where testing was often stuck on at the end of the development process and seen as a process burden rather than a benefit. Testers working in an organization that has adopted agile software development, moved to a DevOps culture, and embraced continuous integration and continuous delivery are therefore required to have at least a basic level of understanding of how to effectively implement test automation as part of their daily activities.

Unfortunately, lots of test automation efforts in agile software development fail, or at least do not maximize their potential.

I'd like to explore what I think are the two most important reasons for test automation not living up to the expectations that testers and other stakeholders in the agile software development process have. Then, let's look at strategies and tactics for how to avoid falling into these traps in order to succeed with test automation in an agile environment.

Unreasonable Expectations

The first of the two main reasons I see so many automation efforts fail to reach their potential is because of the unreasonable expectations that precede its implementation. Too many team leads, development and project managers, and C-level executives (although other roles are not innocent, either) see test automation as the one-stop solution to all of their testing bottlenecks.

However, reality has shown over and over again that:

- Implementing test automation takes time, effort, and specific skills
- Automation is an activity that supports testers, not replaces them
- Far from every activity related to testing can be automated

Yet there's still a widespread belief that test automation somehow, magically and with the press of a button, will perform all of the required testing for you. When, after a couple of months of diligently building and running tests, this notion turns out to be a fantasy, the people who have been working hard on getting the automation to work are often scapegoated, and sometimes even laid off.

In my opinion, the best way testers and automation engineers can deal with this problem is to think and communicate before acting. Make sure that all stakeholders are on the same level with regards to what you can reasonably expect from automation. Take a look at previous efforts, both within your organization and in the wider software testing and development community, and learn from those experiences.

What should work? What will likely not? Don't expect automation to be the magic bullet that will solve all of your testing problems.