AUTOMATED TESTING FOR FASTER INNOVATION

Overview: Manual testing is repetitive and time-consuming. Organizations in the process of adopting DevOps and Agile methodologies can show quick gains in productivity and quality by automating their mainframe testing. By incrementally implementing the right tools and processes, organizations can see significant gains by getting more quality work done with the same resources.

In the modern digital economy, the mainframe, with its long record of security and reliability, is more important than ever before. Increasingly, organizations are looking to incorporate Agile and DevOps on the mainframe to achieve sustained efficient innovation without sacrificing quality. They are finding, though, that traditional manual testing methods are a hindrance to this integration.

Compuware asked Vanson Bourne to conduct a survey of 400 senior IT leaders responsible for mainframe application development at their respective companies. According to these leaders, during the release of new mainframe code over half of a development team's time is spent on testing, with 92% reporting that, because of the increasing complexity of cross-platform application environments, their mainframe teams are spending more time testing than was needed in the past. The majority felt that automation of test cases would reduce the number of bugs in production code and help overcome the growing mainframe skills shortage. In fact, 90% responded that increased

test automation could be the most important factor in their future success. Organizations looking to implement Agile and DevOps, and thus provide continuous value for their customers, can show quick results by reducing waste and conserving resources through test automation.

The Benefits of Automation

A 2019 Forrester Consulting Total Economic Impact™ study showed the benefits realized by a major UK bank when they automated testing with Compuware's test automation tool, Topaz for Total Test. Thanks to a 233% increase in developer productivity and 83% decrease of bugs in production, the bank was able to implement a DevOps culture change and do more work with the same resources. This allowed developers to spend less time on repetitive manual tasks and more time doing work that interested and challenged them, resulting in a reenergized workforce and improved business innovation. As the bank's DevOps transformation manager stated, "Automation should be used to remove some of the tedious parts of the job. No one wants to sit there writing test cases when they could be doing more exciting things like writing code. These tools allow developers to get to the fun stuff over the mundane that the computer can do for you. It's also getting exposure for new technologies. Because the team was so energized and hitting their goals, they were given new things to learn."

First Steps

So, what steps can your organization take to start automating your testing? First, establish baseline metrics for development velocity and number of bugs so you can measure the efficiency and efficacy of your automation. Next, pick a small team that is motivated to learn and adopt test automation.

This pilot team will become familiar with your testing tools, choose a simple program to start with, and create initial test cases. As their familiarity with tooling increases, they will gradually adopt larger programs to test and learn which tools are right for which jobs. They will also familiarize themselves with version control management systems and set up a testing pipeline that can be scaled for larger implementation.

The team will continuously monitor velocity and number of bugs after adopting automation and, based on their experiences, establish best practices and processes. The knowledge gained by this team's initial experiences should be collected into a "Test Automation Knowledge Center" which can be accessed by other teams as your automation program is expanded. It is important to measure adoption of automation across the organization to ensure that bottlenecks do not develop because of particular teams' failure to automate.

Choosing the Right Tools

One key to successful adoption of test automation is choosing the right tools. With more frequent and continuous testing it is important that tests can be run in a virtualized environment that is not dependent upon completion of programs in the production environment or upon the collection of data from an external environment. At some point, though, interaction with other programs and databases within non-virtualized environment will be needed. Topaz for Total Test allows both types of testing.

Virtualized and Non-Virtualized Testing

For quick feedback independent of other developers, administrators, and environments, virtualized testing in a virtualized environment offers more control, allowing a developer to repeat tests with the same manually entered data or perform negative testing.

Non-virtualized testing shows a wider scope of results with more realistic test scenarios, using real data, and allows developers to see how programs interact and detect the impact that a program may have on other programs or on the environment as a whole.

The full benefits of automated testing, though, come not just from individual developers testing, but from process automation and orchestration. Tests must consistently be executed when they should be, without human intervention. Topaz for Total Test assets can be shared using tools like Git, allowing it to be used with highly automated continuous integration tools like Jenkins for test execution and SonarQube for checking results and passing quality gates.

Conclusion

Automated testing provides extraordinary benefits, but implementation can seem daunting. By starting small, learning, iterating, and documenting best practices, and then expanding across your organization, you can reap these benefits and see a marked increase in the quality, velocity, and efficiency of your software delivery.

This post originally appeared on www.gse.org.uk.