Autonomous Testing as Part of Modern Software Assurance and Engineering

Software testing is having a difficult time keeping up with the already frenetic and constantly accelerating pace of releases. According to a 2022 survey from Gitlab, seven out of ten developers said their teams release code at least every few days, with many doing so daily. In this day and age, customers and end-users expect new features and functionality at an increasingly rapid pace. Those companies that fall behind on new software releases will be displaced by their competitors who can keep up with the latest updates.

When testing isn't keeping up with release pace, organizations are faced with the well-known risks associated with releasing software that hasn't been adequately tested and may contain bugs. In July 2022, for instance, former Volkswagon CEO Herbert Diess was forced out of the company because the automaker's software unit was unable to produce software of sufficient quality, delaying the launch of its new Porsche, Audi, and Bentley models. Even more recently, in October 2022, Nintendo had to take Nintendo Switch Sports' servers offline for nearly a week due to a bug that caused the game to crash.

Development teams have tried to address this dilemma—issue potentially buggy software faster or slow down to sufficiently test—with test automation. But there are significant challenges associated with how test automation is traditionally implemented, and automation still requires highly skilled testers, who are always in high demand, so they are difficult to hire and retain.

The challenges that testing organizations face go well beyond the need to automate the creation of tests. There's also the challenge of maintaining tests—otherwise automation scripts end up becoming obsolete so they no longer test the functions they should in the ways required. Even when there are sufficient testers on hand, the task of analyzing the impact of change and configuring the test suite to respond is far too complex to perform manually. But the problem runs even deeper than maintaining automated tests in the face of change. Human analysis is insufficient to identify all the areas that require testing.

The solution is to move beyond testing automation to autonomous testing.

AI-Powered Autonomous Testing

Using autonomous testing enables faster decisions about which scenario to test based on the impact of a change without much human involvement, which dramatically increases testing depth and scope while simultaneously speeding up the process.

Traditional test automation only affects one stage of the testing process: automated script execution in the DevOps pipeline (Figure 1).

Figure 1. The Traditional Testing Process

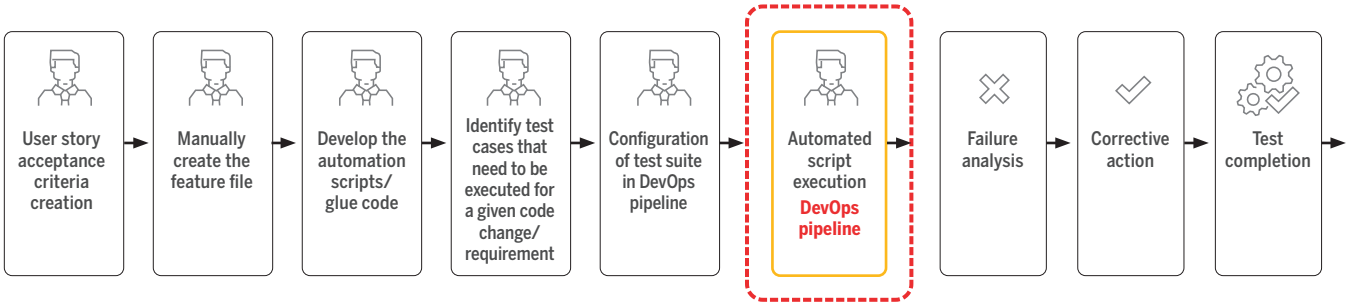
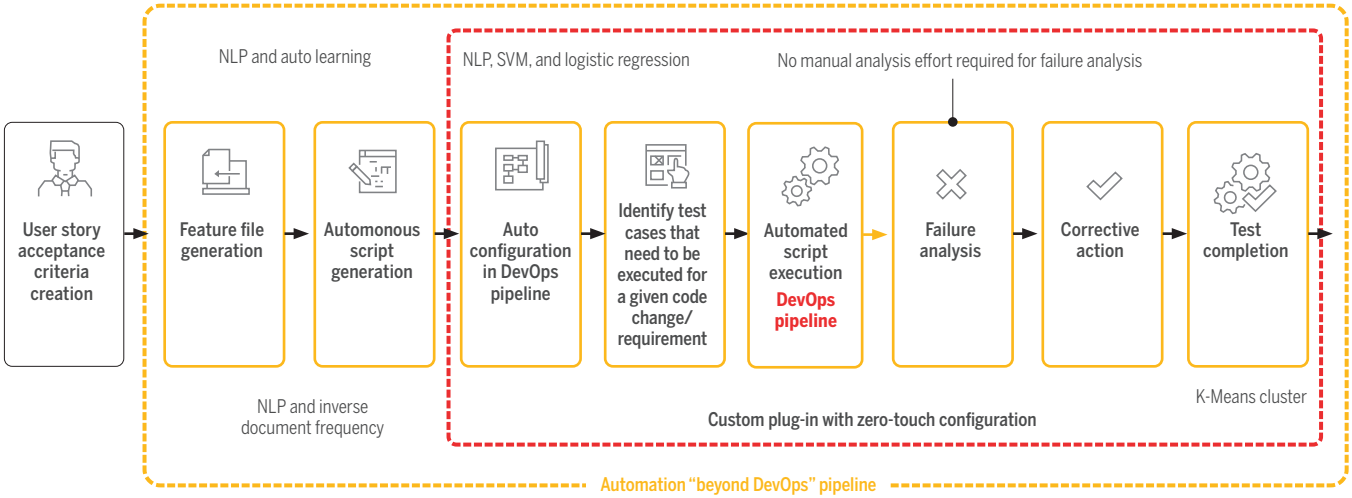


Figure 2. Automation Beyond the DevOps Pipeline



Autonomous testing, however, has the capability to remove and reduce human manual processes from nearly the entire testing process (Figure 2). By applying natural language processing (NLP) and machine learning (ML) technologies, organizations can automate feature file and autonomous script generation. The addition of deep learning via a support vector machine (SVM) can auto-configure tests, identify cases that need to be executed when there's a code or requirements change, conduct failure analysis, and take corrective action.

Over time, the AI continues to learn from development behavior, test results, and other data to get smarter and more accurate. For instance, every development organization produces postproduction logs, but these logs rarely get used. AI can analyze these logs and match them to postproduction to identify previously unrecognized "white spaces" that merit testing because they are likely areas for bugs to emerge in the future.

It is important to note that autonomous testing is not a one-time deployment; it is an ongoing journey that makes progress on a case-by-case basis. It can begin by identifying a specific bottleneck in the process that autonomous testing can address, such as the generation of UI/API scripts or the identification of sensitive columns that need to be masked or replaced with synthetic data. Ideally, whatever the discrete case, it should cut across different functions for a particular phase so that it has a more profound impact. Once that particular case has been implemented and shown results, leverage that success to expand to a new case.

Think of it in terms of autonomous driving. Automakers first rolled out discrete capabilities such as automatic breaking to avoid hitting a stationary object, lane assist, and adaptive cruise control. Implementing autonomous testing requires a similar approach.

Organizations need to conduct more testing in less time with fewer people to produce high quality software on time. Autonomous testing with the assistance of AI and ML, enables organizations to achieve that goal, but they need to take a measured, strategic and long-term approach to implementation. The end result is that development teams can release new features more frequently that improve their customer experience—and the organization's bottom line.

Learn More

Listen to a Thoughtcast that answers key questions about **autonomous software testing** and explains how to move seamlessly from automation to autonomous testing.

Reach us at **marketing@hexaware.com** for more information.

HEXAWARE