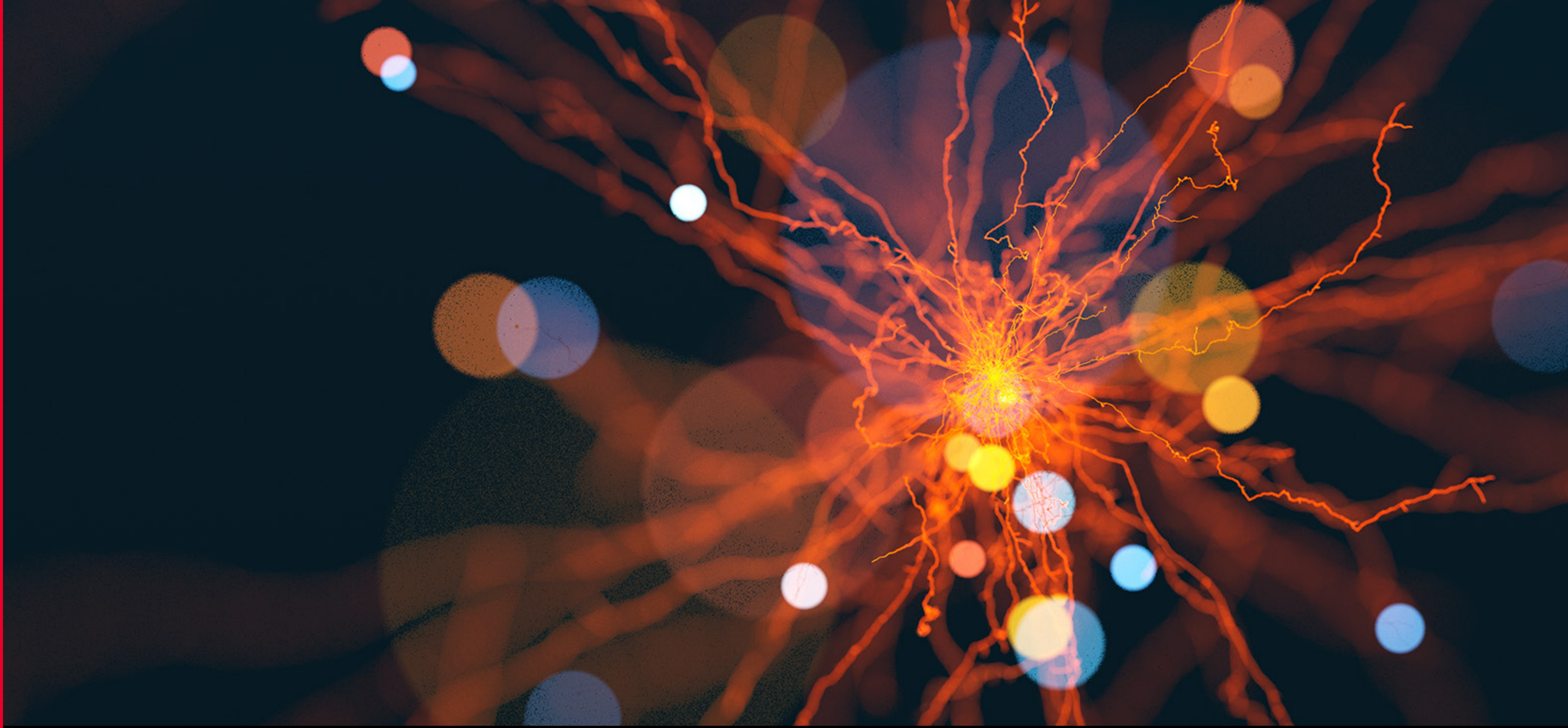


The Ultimate AI Testing Playbook

Intelligent testing, great results

eBook

KEYSIGHT

Overview

To keep up with customer expectations, the pressure to release updates faster is greater today than it's ever been. DevOps and continuous testing should provide the answer to keeping up with the pace of change but unless testing teams are harnessing everything these ways of working have to offer, they aren't enough. The key to success is adding AI testing to the toolkit, as this playbook explains.

Introduction

Modern software development depends on maximizing the speed of release without compromising the quality. This places more pressure on testing teams than ever before. How do they balance these two conflicting requirements?

DevOps and continuous testing are seen as the solution, but many organizations aren't harnessing everything this way of working has to offer, which we see in research that shows us that despite the pressure to increase the speed of release, it's actually slowing down.

This is because the process that sits at the heart of the continuous testing model – automated testing – doesn't provide all the answers on its own. In fact, it often creates more problems than it solves. It's only when you add Artificial Intelligence (AI) to the automated testing toolkit that you open up all the opportunities.

In this playbook, we'll explore the limitations of automated testing and look at best practice options when it comes to adding AI to your toolkit.

According to Forrester Research, release frequency is slowing down. [1] Organizations that leverage an integrated continuous deployment tool set have the advantage, but only 45% of organizations automate release to production. [2]

The Advantages and Limitations of Automated Testing

Continuous testing sits in the software delivery pipeline and gives testing and development teams immediate feedback on the release in development. It's a model that allows bugs to be spotted earlier in the process, making them quicker and easier to fix, which is what's required in a business model where speed of release is a priority.

As the name suggests, continuous testing runs continuously throughout the development process. As such, it isn't feasible for humans to carry out the testing, so this is where automated testing comes in. Automated testing takes redundant, repetitive, manually time-consuming tests and uses a tool to execute these tests without human interaction.

But automated testing is not a silver bullet. There are two key problems with automated testing, especially in a continuous testing environment. In both cases, they relate to the continued need for human intervention.

The high maintenance requirements of automated test scripts

Automated tests are brittle and require considerable maintenance to keep them operational. There are even instances where the support needed to maintain an automated test takes more time than it took to code the function the automated test is testing. This means there is a danger of automated testing holding up a function it was meant to speed up. In practice, what this means is that human intervention is needed on every round of automated testing to perform tests that are awaiting an update. Ultimately, automated testing is only partially automated.

The lack of a gold set in automated testing

An automated testing set looks at two subsets.The first is the regression subset. These are tests that need to be conducted for every release. They check that no previously fixed bug has re-emerged in the latest iteration. The second subset tests new functionality.Problems arise when the testing time frame is too short to run both subsets of tests in full. In this case, the testing manager faces a dilemma: delay the release while the testing is executed in full or make a 'gut feel' decision on which tests need to be run. Delaying the release could lead to a loss of revenue but when the testing isn't completed in full, there is always the risk of a potentially damaging bug being left in a release.

Happily, these problems are well-known – as is the way AI offers the answer. We'll look at how in the next section.

How AI Testing Augments Automated Testing in the Continuous Testing Environment

Test managers are increasingly using AI, machine learning, and deep learning to keep up with the demand and diversity of testing and overcome the limitations of automated testing in a continuous testing environment.

AI works by consuming data, running that data through specific algorithms, and coming up with an optimized set of test cases to execute for any situation. It makes decisions based on those inputs much faster and more accurately than any human can. Crucially, AI doesn't need human interaction to run and can keep working as long as test environments are operational. This opens up exciting possibilities for speeding up automated testing.

It's perhaps worth noting at this stage that we are not in the realms of science fiction where AI does not need humans and can make any decision based on any inputs. Research suggests this type of AI is still at least a century away. [3]

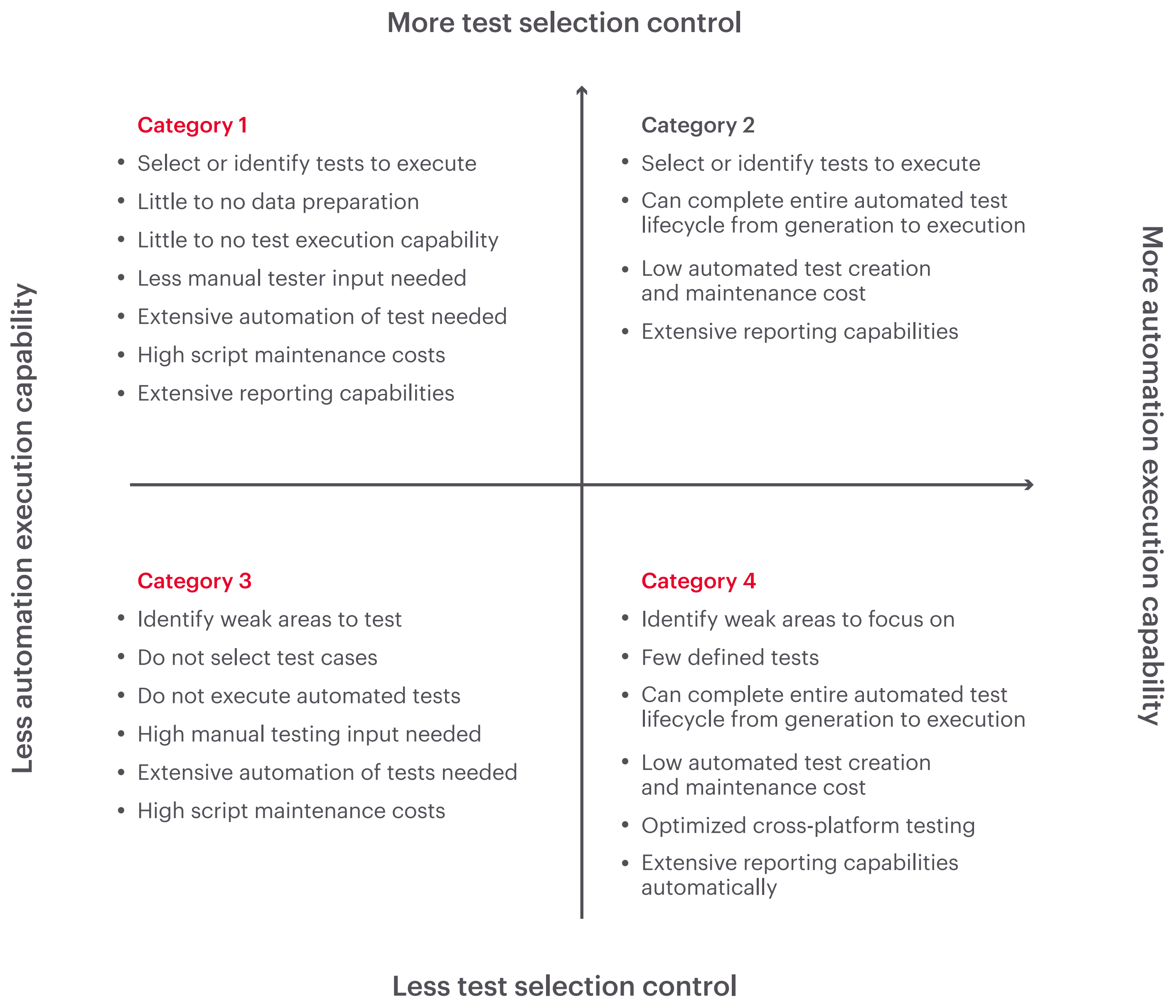
Today's AI still needs to be human guided. This is because it understands algorithms and data but it does not understand human knowledge, context, or emotions. It's perhaps better therefore to think of today's Artificial Intelligence as being Augmented Intelligence. It supercharges human efficiency while at the same time being controlled and directed by the human.

We'll look at how AI is being used in today's testing environment in the next section.

The continuum of AI testing and the products that deliver on it

AI can be used to a greater or lesser extent in the continuous testing environment. For the testing manager, it's a balance between how much automation AI executes autonomously and how much control it has in selecting tests. However, the fact that human input is still required at some level means the testing manager needs to decide how much AI they should implement for the testing to meet release speed and release quality goals.

A number of testing tools have been developed to help testing managers harness what AI has to offer. Broadly speaking, they divide into four categories (see diagram on right) but only the fourth category goes far enough to minimize the risks and maximize the opportunities AI presents. It means organizations need to take care when selecting vendors of AI-powered testing solutions.


More test selection control
Less test selection control
Less automation execution capability
More automation execution capability

Category 1

- Select or identify tests to execute
- Little to no data preparation
- Little to no test execution capability
- Less manual tester input needed
- Extensive automation of test needed
- High script maintenance costs
- Extensive reporting capabilities

Category 2

- Select or identify tests to execute
- Can complete entire automated test lifecycle from generation to execution
- Low automated test creation and maintenance cost
- Extensive reporting capabilities

Category 3

- Identify weak areas to test
- Do not select test cases
- Do not execute automated tests
- High manual testing input needed
- Extensive automation of tests needed
- High script maintenance costs

Category 4

- Identify weak areas to focus on
- Few defined tests
- Can complete entire automated test lifecycle from generation to execution
- Low automated test creation and maintenance cost
- Optimized cross-platform testing
- Extensive reporting capabilities automatically

AI in Test Case Selection

In these tools, AI looks at the existing repository of manual and automated test cases and determines which ones to include for a given release. This use of AI makes the test manager more efficient, but still requires a high level of human input. Why? Because you have to define and maintain your full set of manually agile test cases.

AI in test case execution and error handling

In these tools, AI takes a survey of data and, based on that data, determines which automated scripts to execute. Any automated scripts that fail due to changes in the application under test or within the automated script will be updated and repaired by the AI. This is an exciting advance, but it is still only as good as the data it gets(garbage in, garbage out, as the saying goes), so there is always the risk the fix that has been implemented isn't the right one.

AI in test case selection and execution

These tools combine the first two approaches. Test cases are selected and executed by the AI, replacing the work of the test manager and the test engineer. The need for human input is lessened – but the risk of low quality data or incorrect fixes is amplified. Not having clean data or poor-quality automated tests is just going to replicate the same problems with automation on a much larger, much faster scale.

AI in automation intelligence

The fourth type of tool that uses AI is best described as Automation Intelligence. It takes inputs from several algorithms, and, based on these criteria, chooses which area of the application under test needs to be tested. In an automation intelligence function, the AI looks at several components such as past defect history, application changes, or coverage history to determine what needs testing. The AI then builds the test case, executes the test case, and reports on it.

There are three big advantages of Automation Intelligence. The first is that it does not rely on clean data. Consequently, it does not develop poor-quality automated tests that replicate at scale the same problems with automated testing. And finally, its tests are high quality and robust and can be executed without human intervention. As such, it offers organizations who want to fully embrace DevOps and continuous testing the best route forward.

Eggplant's AI testing harnesses automation intelligence

Eggplant's AI-driven test engine is the best in market for Automation Intelligence. It operates using an ensemble of several algorithms continually fighting for priority and harnesses automation best practices to give testing teams the solution they need.

A democratized testing platform

Eggplant uses a Digital Twin interface. It is a collection of states, actions, and transitions that capture how users use the application under test. This way of working means having domain expertise is more important than having automation expertise. This democratized approach to testing means manual testers and domain experts do not need automation expertise to be able to harness what AI has to offer.

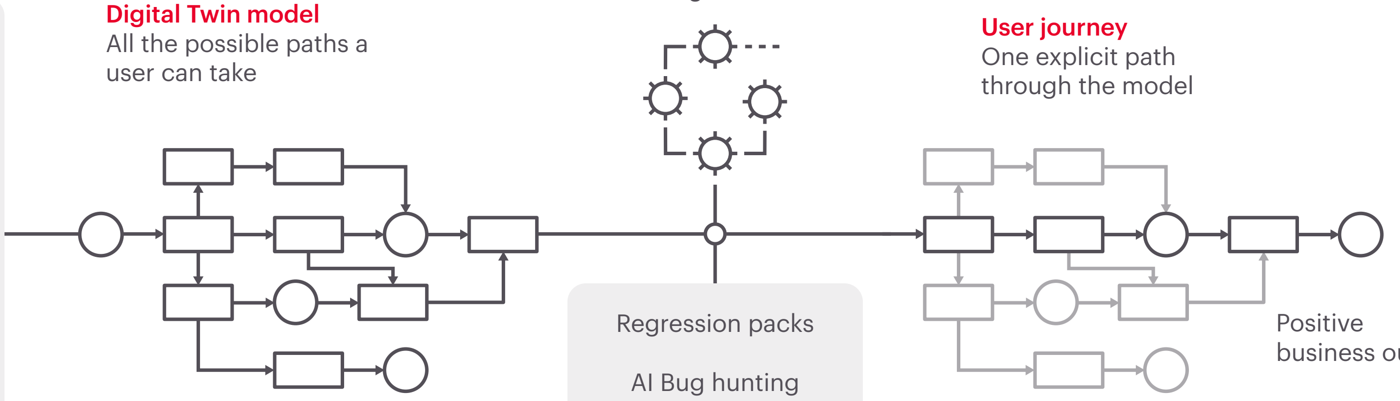
How Eggplant Works

- Interactive model **building**
- Model generation from **existing assets** eg. Gherkin
- Model generation from **real customer** behavior
- Intelligent application **‘scraping’**

Digital Twin model
All the possible paths a user can take

Ensemble of Algorithms
Drive intelligent automated creation of user journeys for testing

User journey
One explicit path through the model



Positive business outcome

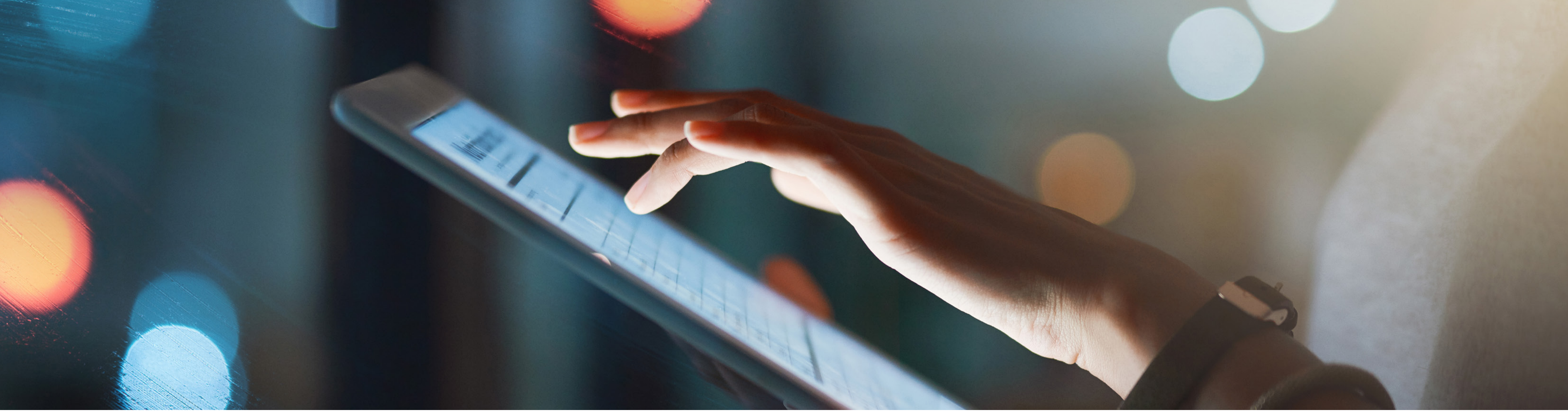
Regression packs

AI Bug hunting

Coverage analysis

Real user journey

Change and defect monitoring



A Best Practice Way of Working

The Eggplant Digital Twin works by using snippets of code associated with the Action or State. These are intelligently stitched together by the AI engine to generate and execute the automated tests.

The snippets of code can be automated scripts or created through Eggplant's natural Sense Talk language, which again democratizes testing. When run, Eggplant AI chooses what it is going to test, builds the automated tests from the code snippets and executes them.

Breaking down larger automated scripts into snippets like this offers three key advantages.

The first is easier test case maintenance. One small script change automatically updates thousands of automated tests, so fewer tests need to be conducted manually while waiting for the automated versions to be updated.

The second is faster test case creation. The AI engine, based on decision-making algorithms, stitches these snippets together to create more significant, more diverse automated tests. This opens up the possibility of running more tests than ever before, even in a limited timeframe.

The third is greater flexibility. The bigger an individual automated script gets, the narrower in scope it gets. Using snippets exponentially increases how that particular bit of code is used in other automated tests. Again, this opens up many more possibilities for tests that can be run.

CASE STUDY

NEC Personal Computers

The mission of the Software Integration Department at **NEC Personal Computers** is to deliver products to the company's customers without any OS problems. Led by Ichiro Mori, PM Group Manager, the team analyzes any defects found by the development or test departments and reproduces and fixes these issues prior to shipping.

Mori states, "**The problems include fatal errors that have an occurrence rate of less than 0.1 percent. Therefore, it's necessary to repeat the same operation more than 1,000 times to reproduce a rare and fatal problem, resulting in an investigation of the cause taking a significant amount of employee hours.**"

NEC Personal Computers was able to automate some of this testing via its proprietary testing tool, but the technology lacked the ability to handle the graphical user interface (GUI) and use keyboard inputs. Mori elaborates, "For example, if any user input like signing in after a restart causes an error, the tool itself cannot reproduce it, resulting in the need for manual intervention and many employee hours of reproduction tests."

Mori knew there was a better approach that could automate this critical user experience testing while also maximizing human testers' time. After evaluating multiple test automation vendors, Eggplant was selected.

NEC Personal Computers was able to quickly begin realizing returns on its Eggplant investment. For example, the company compared the reproduction test of a rare OS error before and after automation with impressive results.

Mori explains, "Before automation, 1,000 trials required 47 employee hours, while this reduced to 21 hours of the testing team's time after automation. A shipping decision in an OS error investigation required 12,000 trials, which means that 553 employee hours for the manual reproduction test can reduce to just 32—resulting in a reduction rate of one-seventeenth." This reduction in manpower is equivalent to a one-year Eggplant license and maintenance fee. As Mori puts it, "This means that our Eggplant investment is covered through a single reproduction test of only one OS error."

Time needed to conduct 1,000 trials

Before AI testing:	**After AI testing:**
47 employee hours	**21 employee hours**

Conclusion

DevOps at scale and continuous testing are fast becoming the only way to test if organizations are to keep up with the pace of change and the frequency of releases users now expect.

AI provides the means to realize all the benefits of continuous testing and deliver at the speed required.

But organizations need to approach AI with care. The wrong solution can create more problems than it solves. On the other hand, the right solution can permanently change business models, create new opportunities, grow revenue and supercharge human productivity.

KEYSIGHT