The Ultimate AI Testing Playbook

From machine learning to agentic AI: Smarter testing for better results

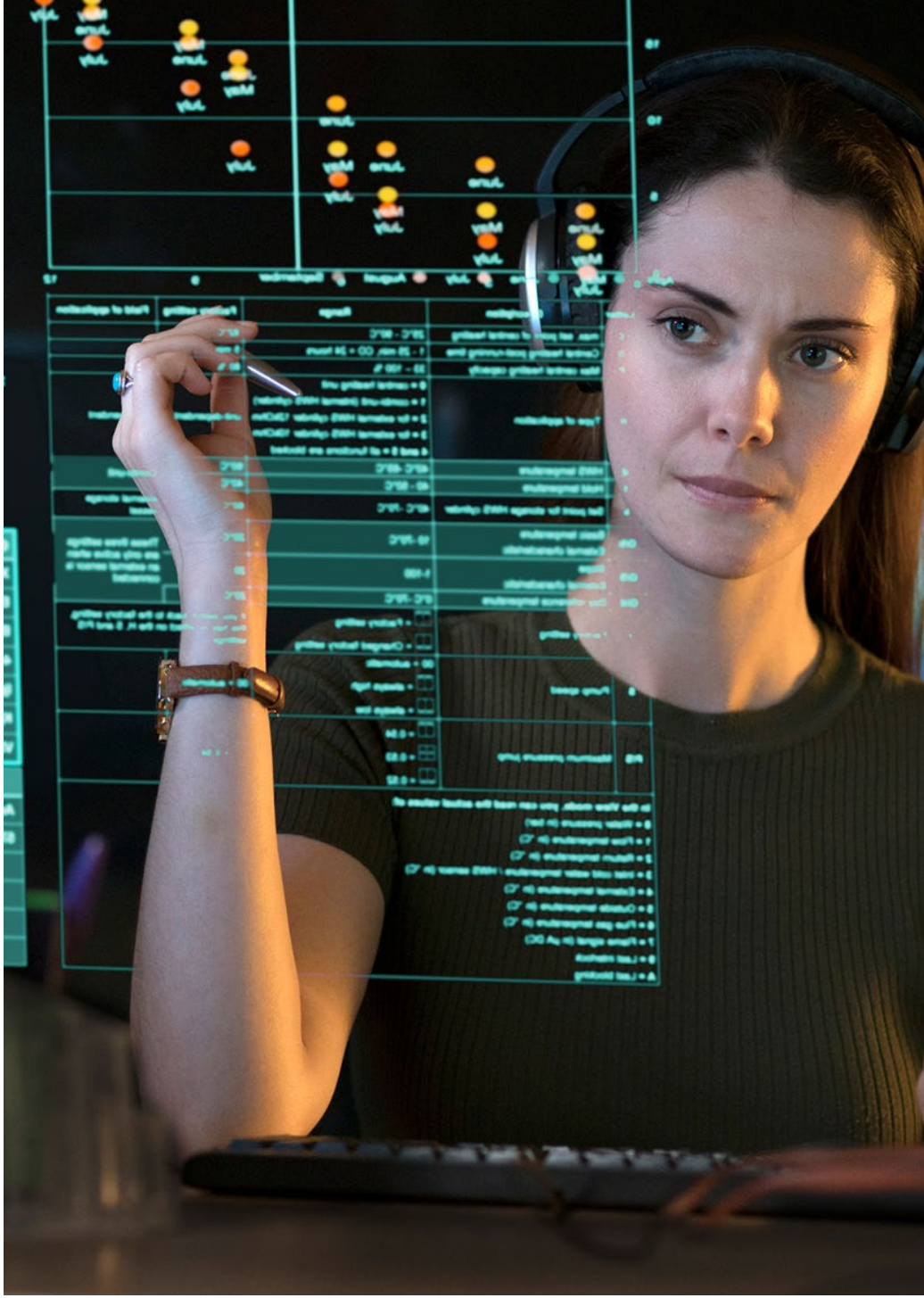
eBook

KEYSIGHT

Overview

The pressure to release updates faster to keep users happy is greater today than it's ever been. DevOps and continuous testing provide part of the remedy to keep up, but unless testing teams harness the latest AI-driven capabilities, they risk falling behind those that do.

Today, AI in testing is evolving beyond simple automation. Generative AI (Gen AI) and Agentic AI are the next advancements, allowing for self-adapting test execution and automated test case generation. Keysight's Eggplant Test has pioneered AI in software testing since 2018, and with elements of these technologies, it continues to push the boundaries of intelligent automation.



Introduction

Modern software development demands that organizations maximize release speed without compromising quality.

However, applications have grown in complexity and rely on multiple integrations and diverse systems, placing significant pressure on testing teams. Testing in isolation is often ineffective, and organizations must contend with fragmented tools, each requiring specific technical expertise.

As continuous testing and DevOps becomes mainstream, many struggle to harness their full potential. For instance, Forrester's Developer Survey in 2024 states that only 23.6% of functional tests, 20.8% of non-functional integrations tests, and 20.5% of end-to-end tests are automated.

By integrating artificial intelligence (AI) into the automated testing toolkit, organizations can unlock unprecedented opportunities. AI-driven testing not only overcomes the limitations of traditional methods but also aligns with DevOps practices, improving key performance metrics such as deployment frequency, lead time, and change failure rates.

In this playbook, we will explore the inherent challenges of traditional automated testing, examine the limitations of current continuous testing models, and introduce best practice options for integrating AI into your testing strategy. The goal is to empower testing teams with the insights and tools they need to balance speed with quality in today's complex software environment.

The Challenges of Traditional Automated Testing

Continuous testing is embedded within modern CI / CD pipelines, providing immediate feedback to development and QA teams. This approach allows bugs to be spotted earlier in the process, making them quicker and easier to fix–an essential requirement for businesses prioritizing rapid release cycles.

As the name suggests, continuous testing runs non-stop throughout the development process, so relying on human intervention isn't practical. As a result, elite software teams use automation to execute repetitive and time-consuming test scenarios without manual interaction.

However, automated testing is not a silver bullet. Three key challenges with automated testing, especially in a continuous testing environment, relate to the continued need for human intervention.

The hidden cost of automated test maintenance

Automated tests can be brittle, requiring significant maintenance. In some cases, the effort to maintain an automated test exceeds the time it took to develop the function it is testing. This creates a situation where automated testing, intended to accelerate development, becomes a bottleneck.

One common cause of excessive maintenance is changes in the Document Object Model (DOM), mainly when object IDs, class names, or element structures are modified. Many automated tests use static locators—such as hard-coded IDs, XPaths, or CSS selectors—to interact with UI elements like buttons, forms, or links. However, when these identifiers change due to UI updates, tests break because they can no longer locate the intended elements. It's often unclear whether a test failure is due to an actual defect or simply a changed ID, forcing testers to update scripts manually, inspect DOM changes, and amend frameworks to restore stability.

In practice, this means that human intervention is required in every single test cycle to address broken locators, rerun failed tests, and manually execute cases awaiting automation updates. As a result, automated testing is often only partially automated, with ongoing maintenance costs that can negate intended efficiency gains.

Testing in isolation and tool integration

Software rarely operates in isolation—most applications rely on multiple integrations, making end-to-end testing complex, especially during regression testing after updates. Organizations often use a mix of specialized tools for user interface testing, API validation, database checks, and code analysis. While each tool serves a specific purpose, this fragmented approach leads to siloed data, inconsistent test scripts, and misaligned environment configurations.

Coordinating these tools introduces challenges in maintaining test coverage, synchronizing updates, and consolidating reports across different platforms. Teams struggle to analyze results efficiently without a unified testing solution, increasing the time required to diagnose failures and maintain quality assurance.

As a result, testers frequently resort to manual workarounds—such as realigning test scripts, manually configuring environments, and consolidating reports—to bridge gaps between tools. These interventions not only slow down the testing process but also undermine the benefits of automation, leading to inefficiencies, higher costs, and an increased risk of human error.

The Conflict Between Intuition and Risk-Based Strategies

An automated testing suite comprises two key subsets: regression tests that run with every release to ensure fixed bugs remain absent and new functionality tests that verify recently added features work as intended. Both subsets are critical: regression tests ensure stability, while new tests drive innovation.

Problems arise when testing time is compressed—common in agile cycles and market-driven schedules. With shortened windows, managers face a dilemma: delay the release to run all tests or rely on 'gut feel' to decide which tests to execute. This intuition-based approach can result in inconsistent coverage and missed edge cases while delays risk revenue loss and competitive setbacks. Studies show that incomplete regression testing leads to significant post-release defects, underscoring the risks of rushing.

Some organizations adopt risk-based testing strategies to mitigate these issues, but even these require time to implement. Fortunately, emerging AI-driven solutions promise to optimize test prioritization and selection. By identifying the most critical tests for each release cycle, AI can reduce manual intervention and balance thorough testing with timely delivery. We'll explore how AI addresses these challenges in the next section.

The Evolution of AI in Software Testing

AI in software testing has advanced through several stages, each adding a layer of intelligence to address increasingly complex testing challenges:

Machine Learning (ML)

Used for pattern recognition, defect detection, and historical test optimization. By analyzing past test results and system behavior, ML AI can identify high-risk areas and automate defect prediction, enhancing test efficiency.execution based on application changes, new feature additions, and detected anomalies, ensuring comprehensive test coverage.

Generative AI (Gen AI)

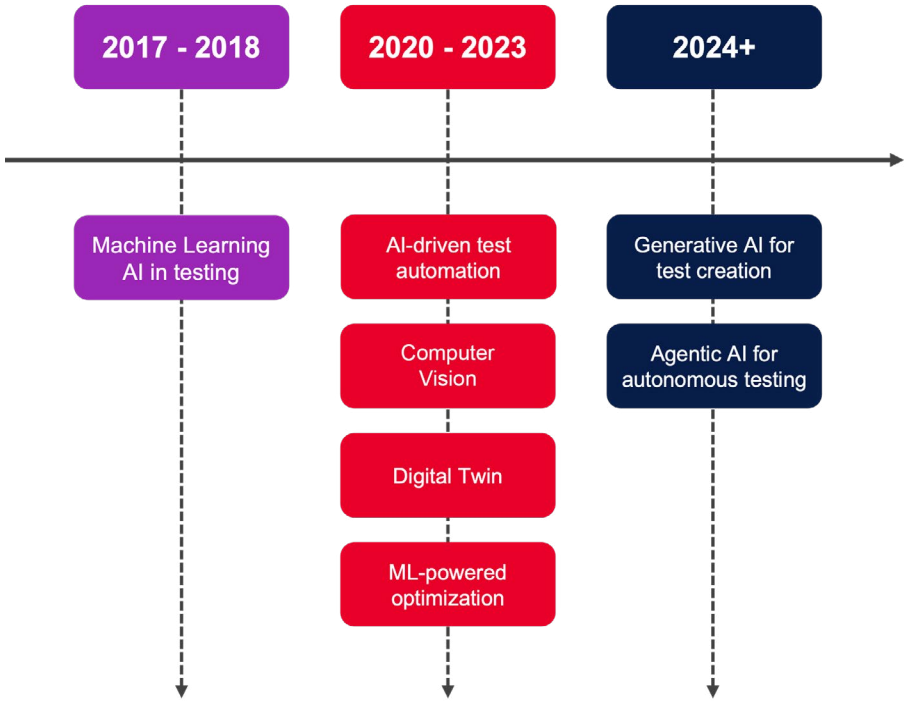
Automates test case framework creation by understanding software requirements and generating test scenarios autonomously. Generative AI can extract information from user stories, code repositories, and historical defects to create optimized, risk-based test cases, reducing manual effort and improving accuracy.

Agentic AI

The latest evolution, where AI can autonomously decide, execute, and adapt tests in real time, minimizing maintenance and human intervention.

Agentic AI has the capability to reason against system behavior, such as Chain of Thought. It can dynamically adjust test execution based on application changes, new feature additions, and detected anomalies, ensuring comprehensive test coverage.

AI Testing Evolution



How AI-Augmented Testing Impacts Continuous Testing

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. Furthermore, AI models are continuously evolving, using vast datasets to improve predictive capabilities and test case prioritization.

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.

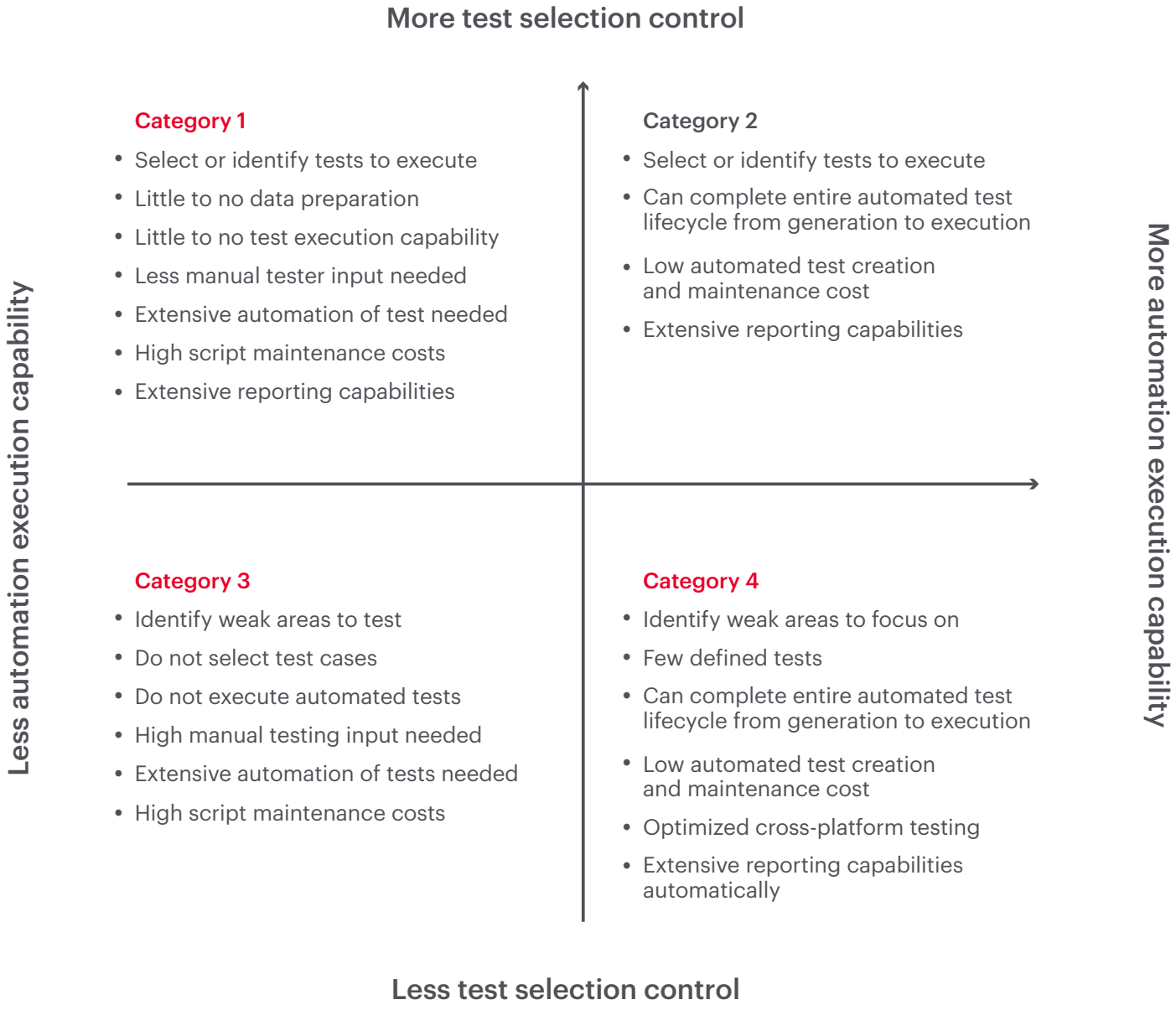
It's perhaps worth noting at this stage that we are not in the realms of science fiction where AI does not need humans and can make any decision based on any inputs. Today's AI still needs to be human guided. This is because it understands algorithms and data, but it does not understand human knowledge, 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.

However, advancements in Generative and Agentic AI are beginning to reshape the way we test. By enabling the generation of test scenarios and scripts from natural language inputs, it looks to bridge the gap between human requirements and machine execution.

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 the next page*) 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
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
Less automation execution capability
More automation execution capability
Less test selection control

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.

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 manual test cases. In addition, Generative AI is beginning to enable testing tools to generate test case frameworks from requirements documentation and natural language, offering more improvements to traditional test case methods.

AI in test case execution and error handling

In these tools, AI takes a reckoning of the system-under-test and associated test data and based on that, 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. Modern AI / ML techniques are being used to improve error prediction, while generative AI could be used to autonomously generate robust error handling routines in the future.

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. Some ML models are being integrated to continuously learn from past test runs and results. Emerging generative AI approaches are enabling test scenario generation that adapts to changing software environments.

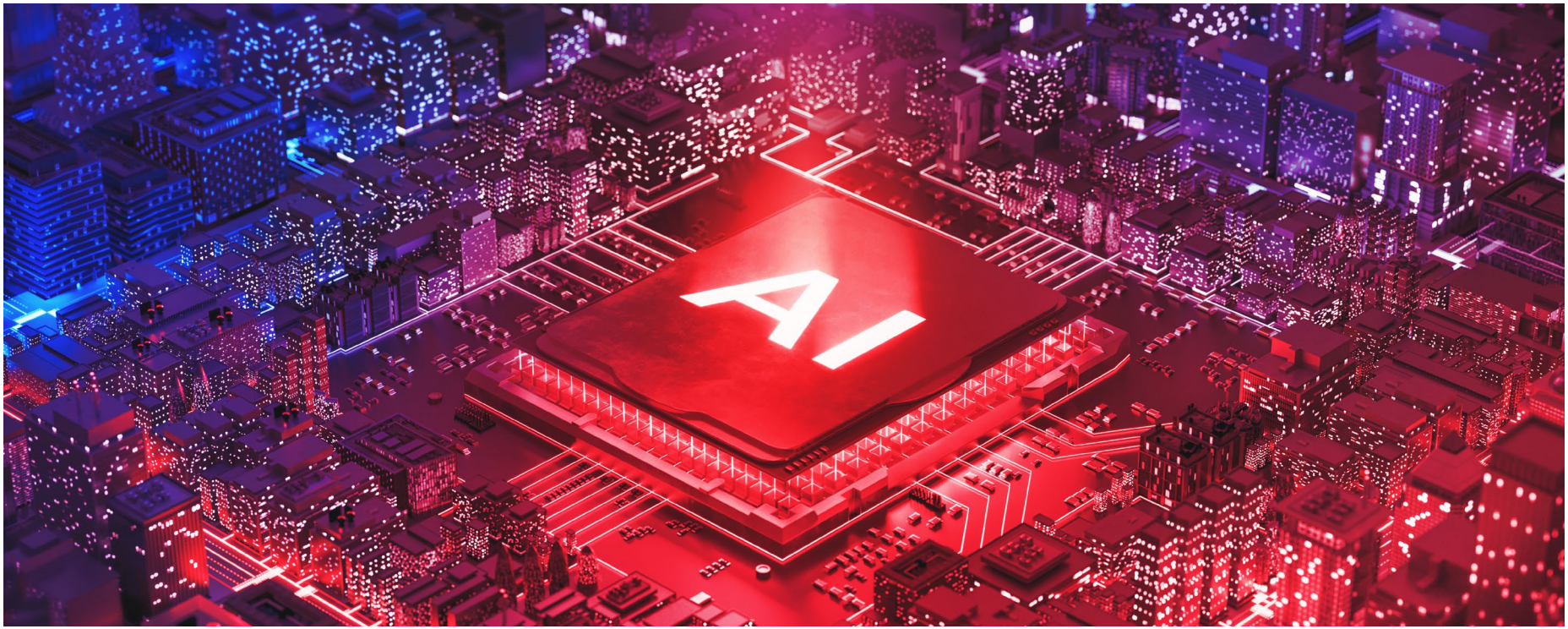
AI in automation intelligence

The fourth type of tool 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, allowing teams to fully embrace continuous testing and DevOps.

Looking ahead, Agentic AI may transform Automation Intelligence by autonomously managing the Quality Engineering Lifecycle—from design to analysis—in real time, paving the way for self-optimizing test environments.



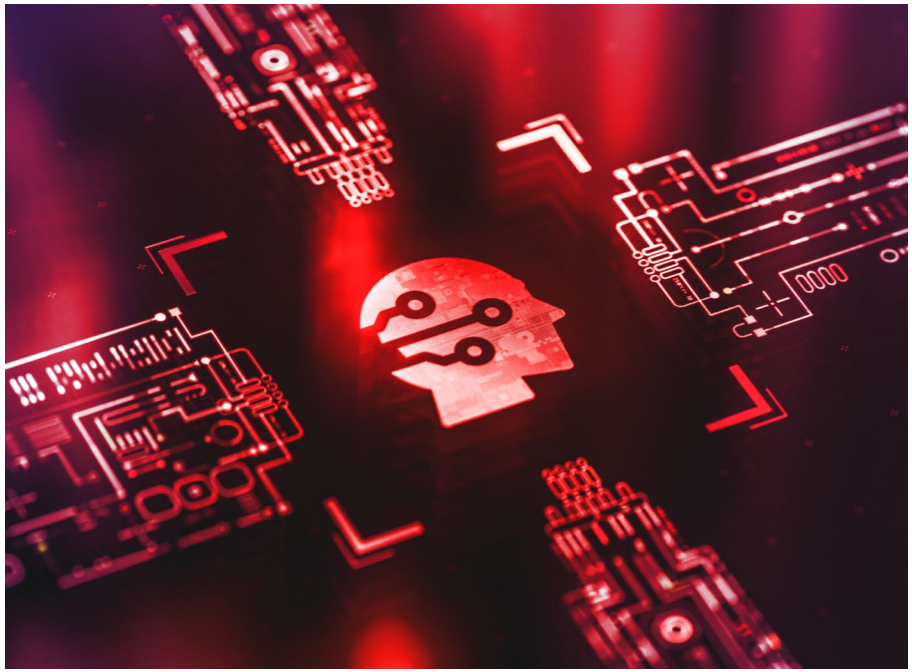
Keysight Eggplant's AI Testing Harnesses Automation Intelligence

Keysight Eggplant's AI-powered 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.

Eggplant already integrates elements of AI, Gen AI, and Agentic AI into single platform. It optimizes test coverage, automates interactions across digital environments, and executes tests just as a human would, all while remaining offline and compliant with AI governance laws in the UK, EU, and US. This is why Keysight Eggplant is the only AI-powered testing solution that prioritizes security, transparency, and governance. Our on-premises approach ensures that all sensitive data remains secure, meeting even the most stringent compliance requirements.

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.

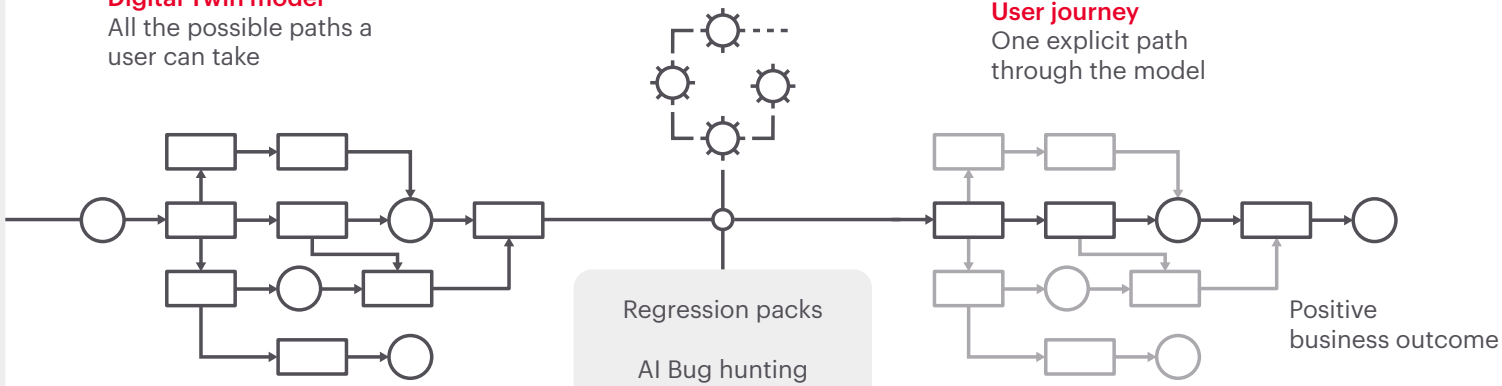


How Eggplant Works

- 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

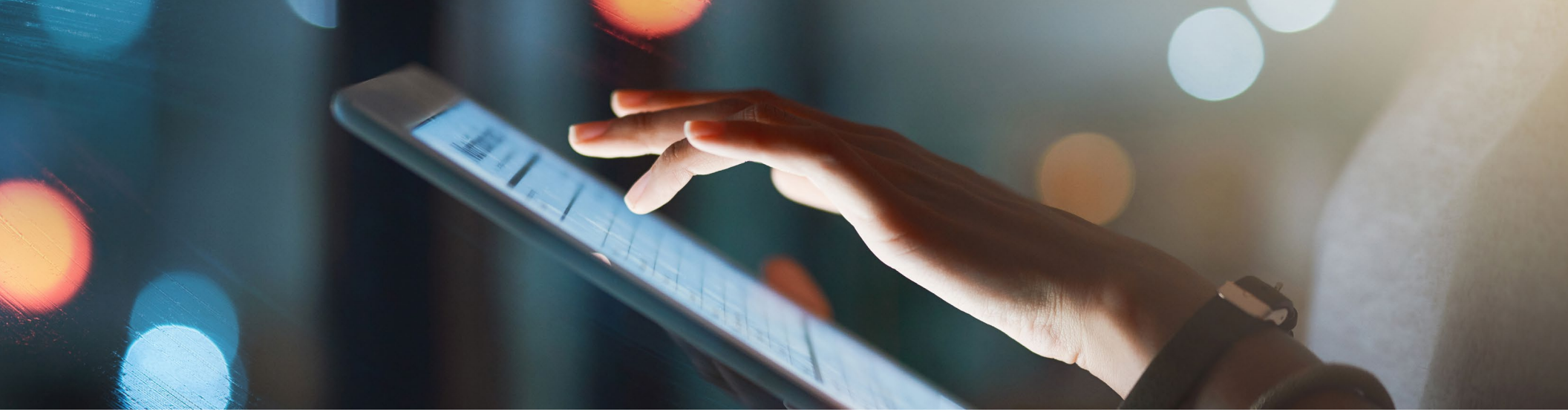
Regression packs

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

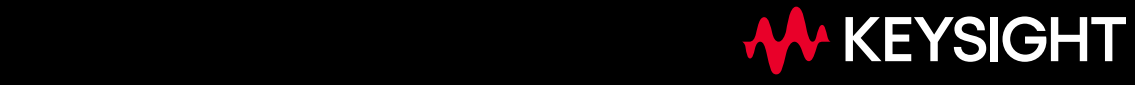
DevOps at scale and continuous testing are fast becoming the only way to test if organizations are to keep up with the pace of change and the frequency of releases users now expect.

AI testing is about more than just following trends—it's about making strategic, future-ready decisions that merge innovation with robust security, scalability, and compliance. AI provides the means to realize all the benefits of continuous testing and deliver at the speed required.

Since 2017, Keysight Eggplant has been a leader in AI-driven testing, developing tools and solutions long before many competitors entered the market. As AI technology evolves, we continue to innovate, ensuring our platform remains at the forefront of secure, offline AI testing.

If you're looking to enhance your software testing approach and need an AI-powered platform that delivers on security, compliance, and flexibility, it's time to explore what Keysight Eggplant has to offer.

Contact us today for a demo or 14-day free trial.



Keysight enables innovators to push the boundaries of engineering by quickly solving design, emulation, and test challenges to create the best product experiences. Start your innovation journey at www.keysight.com.

This information is subject to change without notice. © Keysight Technologies, 2021 – 2025, Published in USA, March 18, 2025, 7121-1201.EN