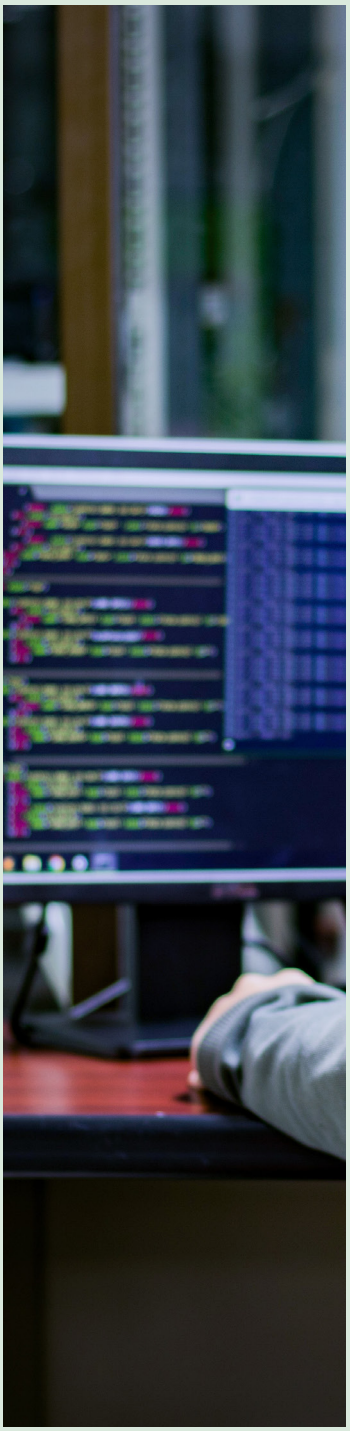
WHITE PAPER



ACHIEVE COMPLETE AUTOMATION WITH ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Abstract

As agile models become more prominent in software development, testing teams must shift from slow manual testing to automated validation models that accelerate the time to market. Currently, automation test suites are valid only for some releases, placing greater pressure on testing teams to revamp test suites, so they can keep pace with frequent change requests. To address these challenges, artificial intelligence and machine learning (AI/ML) are emerging as viable alternatives to traditional automation test suites. This paper examines the existing challenges of traditional testing automation. It also discusses five use-cases and solutions to explain how AI/ML can resolve these challenges while providing complete and intelligent automation with little or no human intervention, enabling testing teams to become truly agile.

Infosys®
Navigate your next

Introduction

Industry reports reveal that many enterprise initiatives aiming to completely automate quality assurance (QA) fail due to various reasons resulting in low motivation to adopt automation. It is, in fact, the challenges involved in automating QA that have prevented its evolution into a complete automation model.

Despite the challenges, automation continues to be a popular initiative in today's digital world. Testing communities agree that a majority of validation processes are repetitive. While traditional automation typically checks whether things work as they are supposed to, the advent of new technologies like artificial intelligence and machine learning (AI/ML) can support the evolution of QA into a completely automated model that requires minimal or no human intervention.

Pain points of Complete Automation

Let us look at the most pertinent problems that lead to low automation statistics:

- Frequent requirement changes – While most applications are fluid and revised constantly, the corresponding automation test suite is not. Keeping up with changing requirements manually impedes complete automation. Moreover, maintaining automated test suites becomes increasingly complicated over time, particularly if there are frequent changes in the application under test (AUT)
- Mere scripting is not automation – Testing teams must evolve beyond traditional test automation that involves frequent manual script-writing.
- Inability to utilize reusable assets – It is possible to identify reusable components only after a few iterations of test release cycles. However, modularizing these in a manner that can be reused everywhere is a grueling task.
- Talent scarcity – Finding software development engineers in test (SDETs) with the right mix of technical skills and QA mindset is a significant challenge

QA teams today are looking for alternatives to the current slow and manual process of creating test scripts using existing methodologies. It is evident that intelligent automation (automation leveraging AI/ML) is the need of the hour.

How can enterprises achieve complete automation in testing?

The key to achieving complete automation lies in using AI/ML as an automation lever instead of relegating it to scripting. Optimizing manual test cases using AI/ML is a good start. Helping the application self-learn and identify test suites with reusable assets can be more advanced utilities for automated test suite creation.

Leveraging AI in test suite automation falls into two main categories – 'automation test suite creation using various inputs' and 'automation test suite repository maintenance'. The following section discusses various use cases for intelligent automation solutions under these two categories which address the challenges of end-to-end automation.

Use cases and solutions for intelligent test automation

A. Automation test suite creation

Use case 1: Optimizing a manual test suite

Testing teams typically have a large set of manual test cases for regression testing, which are written by many people over a period of time. Consequently, this leads to overlapping cases. This increases the burden on automation experts when creating the automation test suite. Moreover, as the test case suite grows larger, it becomes difficult to find unique test cases, leading to increased execution effort and cost.

Solution 1: Use a clustering approach to reduce effort and duplication

A clustering approach can be used to group similar manual test cases. This helps teams easily recognize identical test cases, thereby reducing the size of the regression suite without the risk of missed coverage. During automation, only the most optimized test cases are considered, with significant effort reduction and eliminating duplicates.

Use case 2: Converting manual test cases into automated test scripts

Test cases are recorded or written manually in different formats based on the software test lifecycle (STLC) model, which can be either agile or waterfall. Sometimes, testers can record audio test cases instead of typing those out. They also use browser-based recorders to capture screen actions while testing.

Solution 2: Use natural language processing (NLP)

In the above use case, the execution steps and scenarios are clearly defined, after which the tester interprets the test cases, designs an automation framework and writes automation scripts. This entire process consumes an enormous amount of time and effort. With natural language processing (NLP) and pattern identification, manual test cases can be transformed into ready-to-execute automation scripts and, furthermore, reusable business process components can be easily identified. This occurs in three simple steps:

- Read – Using NLP to convert text into the automation suite.
- Review – Reviewing the automation suite generated.
- Reuse – Using partially supervised ML techniques and pattern discovery algorithms to identify and modularize reusable components that can be plugged in anywhere, anytime and for any relevant scenario.

All these steps can be implemented in a tool-agnostic manner until the penultimate stage. Testers can review the steps and add data and verification points that are learnt by the system. Eventually, these steps and verification points are used to generate automation scripts for the tool chosen by the test engineer (Selenium, UFT or Protractor) only at the final step.

In this use case, AI/ML along with a tool-agnostic framework helps automatically identify tool-agnostic automation steps and reusable business process components (BPCs). The automation test suite thus created is well-structured with easy maintainability, reusability and traceability for all components. The solution slashes the planning and scripting effort when compared to traditional mechanisms.

Use case 3: Achieving intelligent DevOps

Software projects following the DevOps model usually have a continuous integration (CI) pipeline. This is often enabled with auto-deploy options, test data and API logs with their respective requests-response data or application logs from the DevOps environment. While unit tests are available by default, building integration test cases requires additional effort.

Solution 3: Use AI/ML in DevOps automation

By leveraging AI/ML, DevOps systems gain analytics capabilities (becoming 'intelligent DevOps') in addition to transforming manual cases to automated scripts.

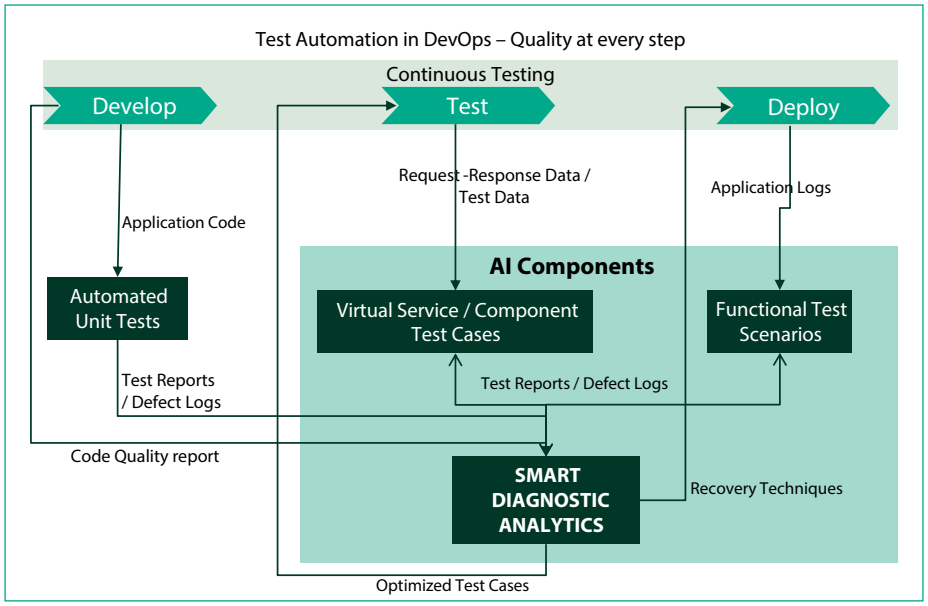


Fig 1: *Achieving complete automation on a DevOps model with AI/ML*

As shown in Fig 1, the key steps for automating a DevOps model are:

- Create virtual service tests based on request-response data logs that can auto update/self-heal based on changes in the AUT.
- Continuously use diagnostic analytics to mine massive data generated to proactively identify failures in infrastructure/code and suggest recovery techniques.
- Leverage the ability to analyze report files and identify failure causes/ sequences or reusable business process components through pattern recognition.
- Enable automated script generation in the tool-of-choice using a tool-agnostic library.
- Analyze past data and patterns to dynamically decide what tests must be run for different teams and products for subsequent application builds.
- Correlate production log data with past code change data to determine the risk levels of failure in different application modules, thereby optimizing the DevOps process.

B. Automation test suite maintenance

Use case 4: Identifying the most critical paths in an AUT

When faced with short regression cycles or ad hoc testing requests, QA teams are expected to cover only the most important scenarios. They rely on system knowledge or instinct to identify critical scenarios, which is neither logical nor prudent from a test coverage perspective. Thus, the inability to logically and accurately identify important test scenarios from the automation test suite is a major challenge, especially when short timelines are involved.

Solution 4: Use deep learning to determine critical application flows

If the AUT is ready for validation (even when no test cases are available in any form), a tester can use a reinforcement learning-based system to identify the critical paths in an AUT.

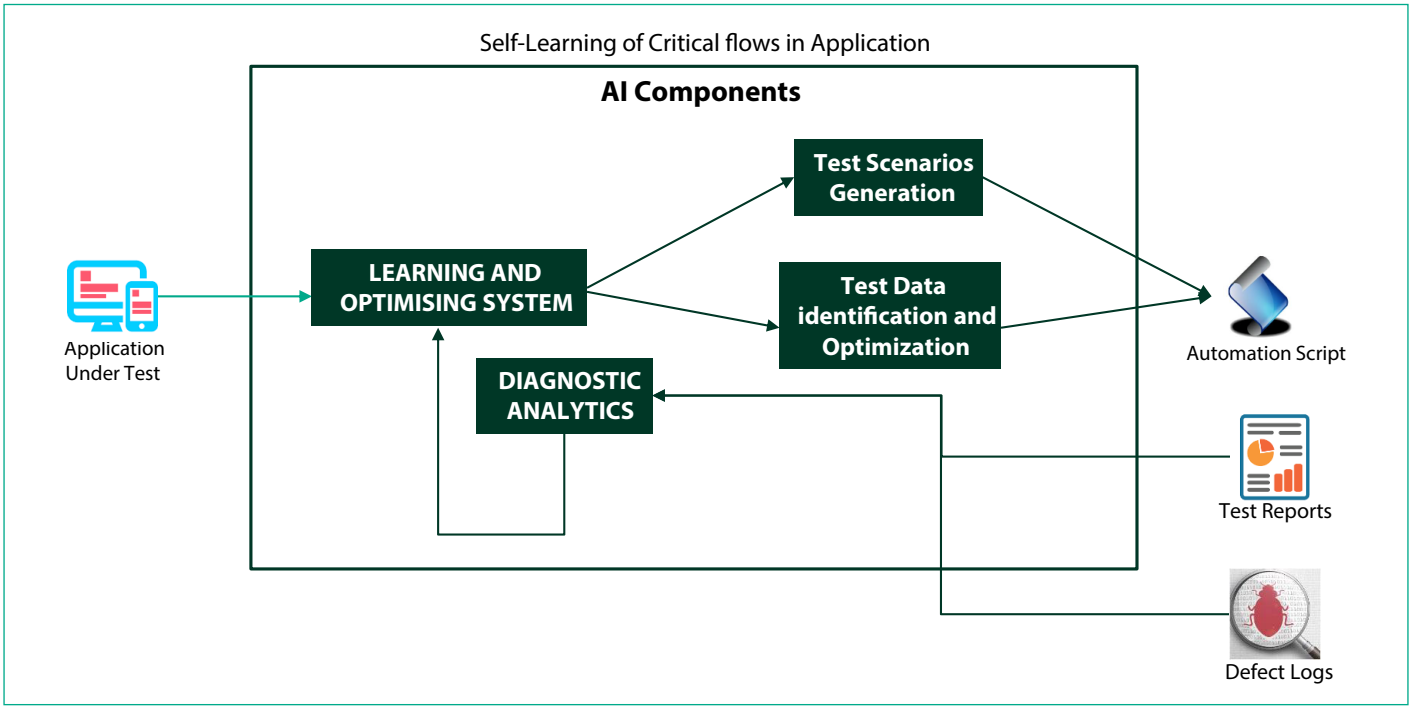


Fig 2: *How reinforcement algorithms in testing identify critical flows in an application*

Use case 5: Reworking the test regression suite due to frequent changes in AUT

If a testing team conducts only selective updates and does not update the automation test suite completely, then the whole regression suite becomes unwieldly. An AI-based solution to maintain the automation and regression test suite is useful in the face of ambiguous requirements, frequent changes in AUTs and short testing cycles, all of which leave little scope for timely test suite updates.

Solution 5: Deploy a self-healing solution for easier test suite maintenance

Maintaining the automation test suite to keep up with changing requirements, releases and application modifications requires substantial effort and time. A self-healing/self-adjusting automation test suite maintenance solution follows a series of steps to address this challenge. These steps are identifying changes between releases in an AUT, assessing impact, automatically updating test scripts, and publishing regular reports. As shown in Fig 3, such a solution can identify changes in an AUT for the current release, pinpoint the impacted test scripts and recommend changes to be implemented in the automation test suite.

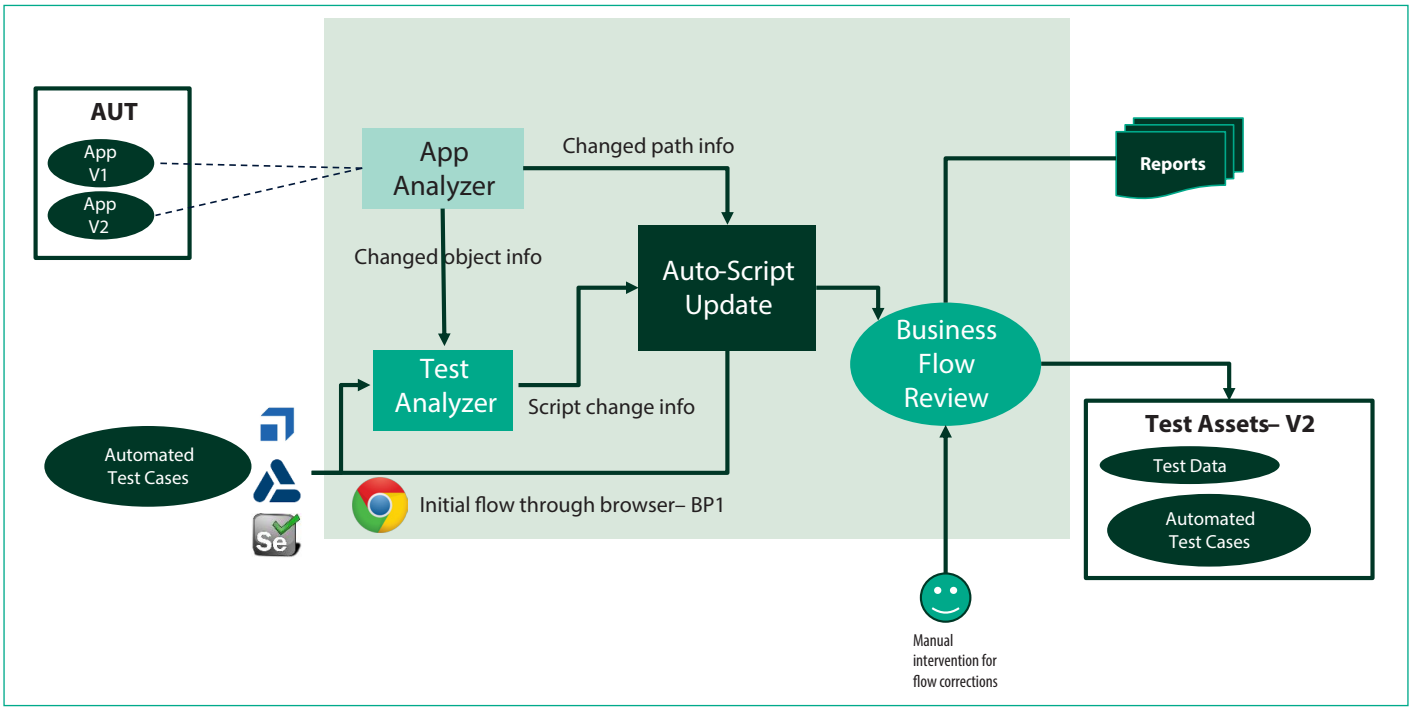


Fig 3: *How a self-healing solution enables faster maintenance of the automation test suite*

The five use cases and solutions discussed above can be readily implemented to immediately enhance an enterprise's test suite automation process, no matter their stage of test maturity.

Conclusion

Complete automation, i.e., an end-to-end test automation solution that requires minimal or no human intervention, is the goal of QA organizations. To do this, QA teams should stop viewing automation test suites as static entities and start considering these as dynamic ones with a constant influx of changes and design solutions. New technologies like AI/ML can help QA organizations adopt end-to-end automation models. For instance, AI can drive core testing whether this involves maintaining test scripts, creating automation test suites, optimizing test cases, or converting test cases to automated ones. AI can also help identify components for reusability and self-healing when required, thereby slashing cost, time and effort. As agile and DevOps become a mandate for software development, QA teams must move beyond manual testing and traditional automation strategies towards AI/ML-based testing in order to proactively improve software quality and support self-healing test automation.

About the Authors

Suman Boddu, Senior Technical Manager, Infosys

Akanksha Rajendra Singh, Consultant, Infosys

For more information, contact askus@infosys.com

Infosys®
Navigate your next