marlabs
driving digital agility

# How Effective QA Automation is Transforming Enterprises in 2022

www.marlabs.com

## TABLE OF CONTENTS



# Overview

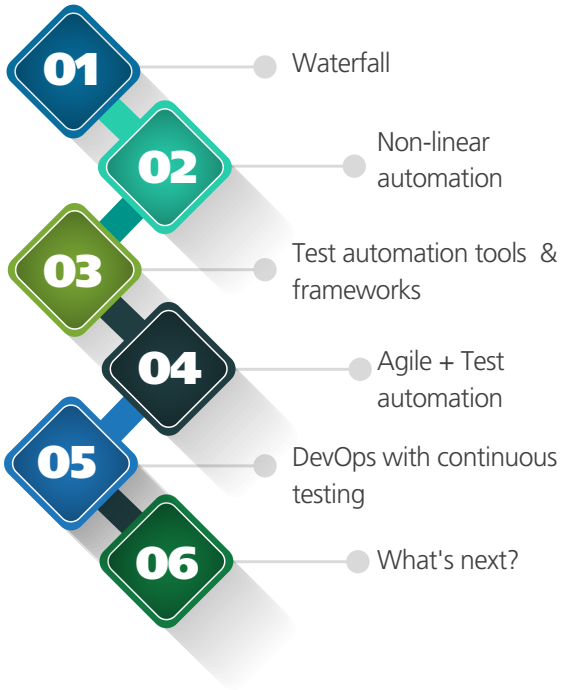
As we step into 2022, there are a lot of opportunities and renewed synergies for growth in the field of QA automation. Organizations aim to improve project efficiencies across the board. There are many trends in quality engineering automation. Today, we'd like to focus on a singular important tenet of QE automation and do a deep dive—the AI/ML space. We will explore some of the areas AI/ML will shine in and some of the benefits it could reap. We will also shed some light on some of the nuances where AI/ML works and where it doesn't.

## How has QA evolved over the years?

In our 20+ years of experience in the technology space, we have seen that the needs and methodologies of projects followed the 'waterfall' model. A lot of these project phases worked serially. The first form of automation arrived in Excel macros and VBAs. That said, the market has evolved over the years. There has been a solid shift to the Agile methodology over the last decade. While the transition to Agile was underway, much focus and maturity were shifted to test automation that leveraged data-driven, keyword-driven, and hybrid-driven frameworks.

At the same time, the steps that the automation initiatives had to take were significant. In the Agile phase, the adoption of automation has increased significantly. As a result, the coverage of test automation has increased too. With the evolution of DevOps in the last 5–10 years and the adoption of CI/CD in delivery, most current projects require bespoke test automation approaches. It must be explored so that testing can be completed within set timelines. This has led to various test automation tools coming to the fore. Emerging technologies have helped QA automation evolve and adapt to the growing technology needs.

### Acceleration in the QA Space




- Speed shouldn't compromise quality
- Testing has to be smart enough with growing technology landscape

# Latest trends in QA automation

From a tech adoption and test automation standpoint,

- **Digital adoption**—refers to the advancements of 'digital' and the proportionate changes in the test automation field. In the last two years, post-Covid, much focus has led industries to adopt digital and become digitally savvy. The urgency to shift to digital has made the testing phase vital. i.e., any applications built must be a multi-form factor and must be created and tested across multiple channels to yield a better omnichannel experience. A deep digital focus strategy is equally important—the 'n' and 'n-1' versions on mobile hardware and software are predominantly what is being focused upon. Most applications nowadays are 3-tier or multi-tiered, and there is much involvement of microservices and containerization. Keeping this in mind, in the middle tier levels, APIs, services, etc., is not just on the UI. More focus is on API testing, backend database, and data testing. Automation has evolved beyond user interfaces, and the onus is now on tiered application-driven automation. This is a trend/norm now that every tester must focus on. Many great tools on the mobility and API sides and robust platforms like Tosca can work across all applications. On the functional side, there are other areas where automation must be adopted, such as usability aspects of the application, customer experience in terms of how an end-user would look at it, what sort of testing and automation can be done, and accessibility (for people who have trouble going through websites).

- **JavaScript-based front-end frameworks**—standard .NET-based applications in the past were very different from those in use today, especially in Java-based scripting. The frameworks have evolved too. Newer Java-based frameworks have led to a more application-based focus with the modern web.

  Examples of such frameworks are: Protractor (Angular), NightJS (AEM), Cypress (JS) etc.

- **Robotic process automation (RPA)** resides on the correction layer, and many organizations leverage RPAs in test automation. Some new players focus on tool rationalization and consolidation. There is a strong link around leveraging RPA on the test automation side. While these might be technological advancements, what happens at the backend for test automation depends on how the automation is being built.

  Most of the prominent frameworks (keyword-driven/data-driven/procedural)—someone would essentially have to do the coding. This led to a barrier of who could only contribute to automation and who could be an automation champion. Script-less automation has emerged in the last couple of years to eliminate this barrier and bring collaboration and automation to business users and testers. This automation helped the business users and teams be part of test automation (including non-programmers).

- **Speed vs. quality**—with the adoption of tools, technologies, and methodologies, test automation has grown significantly. At the same time, the timelines to release applications have shrunk tremendously.

# Top challenges in test automation

There has been tremendous improvement in automation coverage levels—adoption and test automation practices. These two are a must-have in all projects right at the beginning. However, implementing test automation across an enterprise remains a difficult challenge.

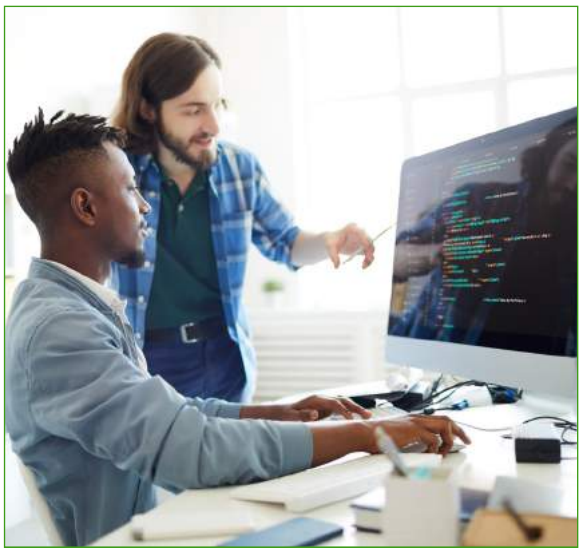

- **Lack of test type and tester support**—medium/large enterprises have existed for 20–30 years. They have developed various technologies around mainframe tech, desktop applications, responsive web apps, mobile apps, IoT-enabled technologies, etc.

- **Slow developer feedback**—speed of input given to the dev team is vital. In a typical Agile sprint, one might have to deal with around 10–12 builds in a 2-week sprint. During this time, teams need to pass on timely feedback about application quality by running tests daily. Many times, there are challenges around this. Primarily because the test automation levels are low, and even on the higher side, they often aren't integrated with DevOps plans. It becomes challenging to run unattended tests during the build.

- **Inability to generate insights**—much data is collected as part of testing and made available for the respective teams. However, dashboards created around this data rarely provide valuable insights to a tech architect/test manager to help them decide when a product should be released. As a result, the insights aren't being derived meaningfully.

- **Inability to identify impact** is very hard for developers to write unit tests during a two-week sprint. As a result, the expectation is that testers should find 100% of the defects during testing. This is an over-reliance on the testing team, and the result is often a prolonged and inefficient feedback loop. Most defects are logged towards the end of the test cycle, and the user stories don't move into completion due to many bugs. Some organizations have achieved very high automation coverage for unit testing and on-time testing. Thousands of regression tests made available in these companies can complete testing in just a few days. However, it is often difficult to understand what is to be tested when a new build is given to the testing team. Very often, teams take calculated guesses with risk-based testing. This results in passing on defects to production teams.

- **High maintenance and failure analysis effort**—in organizations with high volumes of UI-based automation scripts, one of the most common challenges observed is that many scripts fail during automation test cycles primarily due to changes in the application.



Several changes are happening to UI technology, too, with the onset of various new, updated frameworks. Due to this, UI automation scripts depend on web pages and mobile screens. So, many test scripts fail when these pages are updated, and one tries to run them during a formal regression cycle. And time gets wasted trying to fix these problems and re-running the regression tests, and, in some cases, manual testing might have to be taken up too. On average, testing teams spend around 20–25% of their time allotted to test execution on failure analysis.

# How do we address the challenges of QA automation?

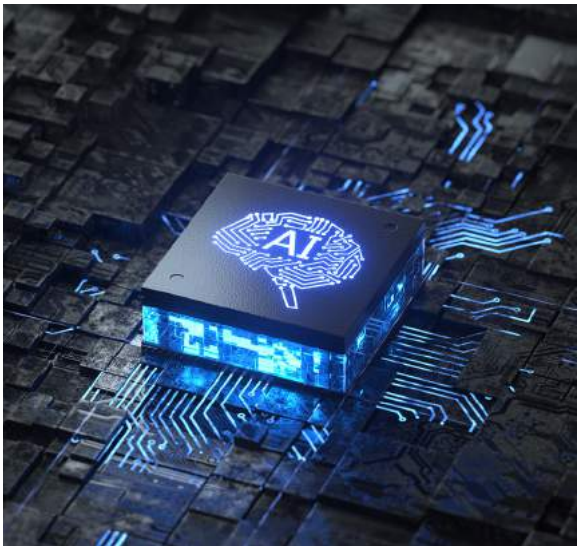
Artificial intelligence can augment engineers with intelligence.

- **Platform approach**—this is platform-enabled quality engineering. A platform-based approach over multiple tools in an organization would be more beneficial. This is very relevant in today's world. Today, numerous tools exist around test automation and different types of testing. It is challenging for tech teams to move from one individual tool to another as the user interface may vary, the approach to automation is different, and so on. A singular platform like Tosca can be used be for different types of testing, automate other technologies, automate legacy applications, etc. This could also be a good bet for end users. Platform-based approaches bring in better reporting. All capabilities around functional and non-functional testing can feed in data to an analytics/reporting platform from where we can derive valuable insights and take decisions.

- **Intelligent test generation**—today, many AI-based solutions can help you generate tests at a much faster pace. E.g., let's consider a piece of Java code that a developer has written. Some tools have the intelligence to go through the code logic and generate unit test cases. This dramatically improves developer coverage and can save much time that would otherwise have been spent on the generation of dev tests. In addition, this can be applied to end-to-end tests—to generate the optimal number of test cases.

- **Autonomous testing**—how do we reduce and optimize tests? Using crawling bots that are automation scripts made to pass through an application. The bots have the intelligence to go through every application screen and identify the workflow based on algorithms. This can address compatibility issues across different mediums such as web, mobile, etc. In addition, autonomous bots are great for reducing the timeframes of the feedback given to the dev team.

- **Predictive analytics**—Tableau and PowerBI are great for dashboards. Algorithms can be used for generating simple insights. For example, you could predict the number of defects. You could develop the top 100 tests in terms of cost, quality when testing should be stopped, etc. The importance of this will get better over time.



- **Change analyzer**—it would be great to have a solution that analyses application changes automatically and thereby helps to reduces maintenance overhead and test failure rates.

# QA automation with AI in action

How does AI allow us to create tests that can keep up with the speed and time-to-market? What are some of the revolutionary things we can do with features such as self-healing? One of the good things about AI tech is it is intended to solve a specific problem (speed, stability, simplicity). This helps ensure that we're putting our minds to solve the best problems rather than getting stuck with scripts.

# What is Vision AI, and how does it work?

Vision AI is a real-time object tracking engine. It picks out every piece of information on a screen and catalogues it. Vision AI can look at a screen and accurately predict the interaction. We can stub things out even before the dev team starts their work. Vision AI can create a module based on a wireframe or module. When designed early in the cycle, AI testing tools can build tests quickly in the cycle (even before the UI is created). We can declutter the space we get into when we try to push features to production as fast as possible. We can stay apace with DevOps groups and sprints by getting in early and designing these tests before.

# Self-healing AI

We should aim to write super resilient tests. Using Tosca, we can tell the tool to look at a test and make observations. The next time the test runs, something may have changed in the feature. A developer may have pushed a change. Usually, even a minute change can break testing cycles, and one would have to go back, recode, fix the module, etc. However, with Vision AI, we can have Tosca figure this out for us and bridge the gap. This will help build on resilient tests that are immune to small changes and doesn't break—this reduces roadblocks to testing and iterations.

# Tosca self-healing AI

Tosca's self-healing AI automatically recognizes new object identifiers as applications change, leading to more stable, resilient, and faster test automation. It is available for both the TBox and Vision AI engines.

# Takeaway?

Marlabs has deep quality engineering expertise. Our partnership with Tricentis as a strategic partner will ensure that we provide the best quality engineering solutions and continuous testing initiatives that your organization might need.


George Ukkuru
Technology Principal
QE & Automation
Marlabs

Siva Prasanna Vanapalli
Associate Vice President
Quality Engineering & DevOps Practice
Marlabs

# About Marlabs

Marlabs designs and develops advanced digital solutions that help its clients improve business outcomes swiftly and precisely. It succeeds by harnessing the power of the Digital Collective™, which brings together design-led digital innovation with human experience, composable digital platforms, and a collaborative ecosystem of frst-class technology partners and innovators.

Marlabs is headquartered in New Jersey, with offices in the US, Germany, Brazil and India. Its 2500+ global workforce includes highly experienced technology, platform, and industry specialists from the world's leading technical universities.

Marlabs Inc.(Global Headquarters) One Corporate Place South, 3rd Floor, Piscataway
NJ - 08854-6116, Tel: +1 (732) 694 1000 Fax: +1 (732) 465 0100, Email: contact@marlabs.com


facebook.com/marlabsinc


linkedin.com/marlabsinc


twitter.com/marlabs


youtube/marlabsinc