Tricentis

5 reasons enterprise test automation is so challenging

INTRODUCTION

In yesterday's waterfall development iterations, manual testing was often a viable—though costly—solution. The benefit of automation has always been evident. Yet, with labor arbitrage, the low cost of manual testing allowed it to remain prevalent for much longer than it should. With cost-effective manual testing options at their fingertips, organizations deferred initiatives to build and scale test automation.

Even 5 years ago, only 30% of enterprise software testing was performed fully "in house," and the vast majority of that testing was not automated. Today, 97% of organizations are practicing Agile to some degree and 73% have active or planned DevOps initiatives. With this fundamental shift, test automation reaches a tipping point. Testers are expected to be embedded in the team so that testing can be completed "in-sprint."

Why does the shift to Agile and DevOps make test automation imperative?

- **There's less time available to test:** Modern application development involves releasing increasingly complex and distributed applications on increasingly tight timelines. It's just not possible to complete the required scope and complexity of testing "in sprint" without a high degree of automation. Manual test cycles take weeks. This doesn't align with today's cadences, where 2-week sprints and (at least) daily builds have become the norm—and the trend is now edging towards Continuous Delivery.

- **Teams expect continuous, near-instant feedback:** Agile and DevOps teams expect feedback to be delivered continuously throughout the release cycle. This just isn't possible with manual testing—even if you hire an entire army of manual testers (which would be exorbitantly expensive, by the way). Without fast feedback on how the latest changes impact core end-to-end transactions, accelerated delivery puts the user experience at risk with each and every release.

- **Business expectations are dramatically different:** As companies prioritize Digital Transformation initiatives, the old adage of "speed, cost, quality...pick two" no longer applies. Amidst pressure to stabilize (and even reduce) costs, IT leaders are now expected to deliver more innovative applications faster than ever. Today, everyone from the CEO down recognizes that skimping on quality inevitably leads to brand erosion as well as customer defection. In regulated industries, the repercussions of subpar quality are even more severe.

Most organizations already understand that test automation is essential for modern application delivery processes. They're just not sure how to make it a reality in an enterprise environment—without exorbitant overhead and massive disruption.

You can't really blame them. Although there's no shortage of test automation success stories floating around software testing conferences, webinars, and publications, they primarily feature developers and technical testers that 1) are focused on testing simple web UIs and 2) have had the luxury of building their applications and testing processes from the ground up in the past few years. Their stories are compelling—but not entirely relevant for the typical Global 2000 company with heterogeneous architectures, compliance requirements, and quality processes that have evolved slowly over decades.

TEST AUTOMATION REALITY VS. TARGET

Before diving deeper into test automation, let's clarify what we're talking about here. Many types of tests can (and should) be automated. For example:

- **Unit tests** that check a function or class (programming units) in isolation
- **Component tests** that check the interactions of several units in the context of the application
- **Functional validation tests** that determine whether a specific requirement is satisfied
- **End-to-end functional tests** that exercise "end-to-end" business transactions across multiple components and applications from the user perspective (UI or API layer)
- **Performance tests** that measure an application's reliability, scalability, and availability under load—at any of the above levels

This paper focuses on functional validation and end-to-end functional tests. Yet, most reports of "test automation rates" include all types of test automation—including unit test automation, which is commonly practiced by developers.

At Tricentis, we've found that companies initially report that they have automated around 18% of the end-to-end functional tests they have designed and added to their test suite. It's actually much lower when you consider how many tests are actually running on a regular basis. And, when you focus on Global 2000 enterprises, it drops even further to a dismal 8%.[1]

The World Quality Report, which is based on 1,600 interviews drawn primarily from companies with 10,000+ employees, also reports test automation rates below 20%:

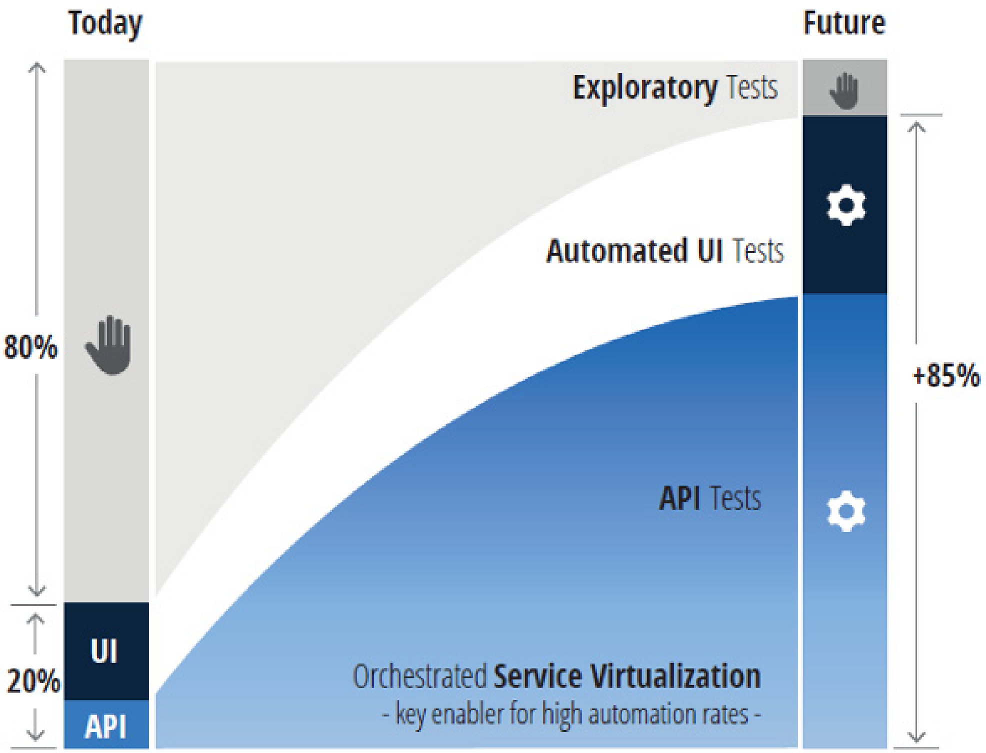
"The level of automation of testing activities is still very low (between 14-18% for different activities). This low level of automation is the number-one bottleneck for maturing testing in enterprises."

Whichever source you choose, the bottom line is the same: there's a huge gap between where we are and where we need to be.

Where should we be? Forrester Research reports that Continuous Testing requires test automation rates to be much higher: "As a rule of thumb, manual testing should account for less than 20% of the overall testing activity; automated testing should account for more than 80%."

[1] Tricentis research conducted from 2015-2018.

This leads to what I call "the Continuous Testing rainbow":


Today
Future
Exploratory Tests
Automated UI Tests
API Tests
Orchestrated Service Virtualization
- key enabler for high automation rates -
80%
20%
UI
API
+85%

To enable Continuous Testing, automation rates needs to exceed 85%. The only remaining manual tests should be exploratory tests, and the type of automation should shift as well. Automation should focus predominantly on the API or message level, requiring service virtualization to simulate the many dependent APIs and other components that are not continuously available or accessible for automated end-to-end testing. UI test automation will not vanish, but it will no longer be the focal point of automation.

Before exploring what's needed to reach this ideal state, let's take a look at why there's such a gap in the first place.

Many enterprise (e.g., Global 2000) organizations have experimented with test automation: typically, automating some UI tests and integrating their execution into the Continuous Integration process. They achieve and celebrate small victories, but the process doesn't expand. In fact, it ultimately decays. Why? It usually boils down to roadblocks that fall into the following categories:

- **Time and resources**
- **Complexity**
- **Trust**
- **Stakeholder alignment**
- **Scale**

Time and resources

Teams severely underestimate the time and resources required for sustainable test automation. Yes, getting some basic UI tests to run automatically is a great start. However, you also need to plan for the time and resources required to:

- Determine what to test and how to test it
- Establish a test framework that supports reuse and data-driven testing—both of which are essential for making automation sustainable over the long term
- Keep the broader test framework in sync with the constantly-evolving application
- Execute the test suite—especially if you're trying to frequently run a large, UI-heavy test suite
- Review and troubleshoot test failures—many of which are "false positives" (failures that don't indicate a problem with the application)
- Determine if each false positive stems from a test data issue, a test environment issue, or a "brittle" script (e.g., a test that's overly-sensitive to expected application changes, like dynamic name and date elements)
- Add, update, or extend tests as the application evolves; the more "bloated" your test suite is (e.g., with a high degree of redundancy and low level of reuse), the more difficult it will be to update
- Determine how to automate more advanced use cases and keep them running consistently in a Continuous Testing environment
- Review and interpret the mounting volume of test results

Complexity

It's one thing to automate a test for a simple "create" action in a web application (e.g., create a new account and complete a simple transaction from scratch). It's another to automate the most business-critical transactions, which typically pass through multiple technologies (mobile, APIs, SAP, mainframes, etc.) and require sophisticated setup and orchestration. To realistically assess the end-to-end user experience in a pre-production environment, you need to ensure that:

- Your testing resources understand how to automate tests across all the different technologies and connect data and results from one technology to another

- You have the stateful, secure, and compliant test data required to set up a realistic test as well as drive the test through a complex series of steps—each and every time the test is executed

- You have reliable, continuous, and cost-effective access to all the dependent systems that are required for your tests—including APIs, third-party applications, etc. that may be unstable, evolving, or accessible only at limited times

Trust

The most common complaint with test results is the overwhelming number of false positives that need to be reviewed and addressed. When you're just starting off with test automation, it might be feasible to handle the false positives. However, as your test suite grows and your test frequency increases, addressing false positives quickly becomes an insurmountable task.

Once you start ignoring false positives, you're on a slippery slope. Developers start assuming that every issue exposed by testing is a false positive—and testers need to work even harder to get critical issues addressed. Moreover, if stakeholders don't trust test results, they're not going to base go/no-go decisions on them.

Stakeholder alignment

Continuous Testing is all about providing the right feedback to the right stakeholder at the right time. During the sprint, this might mean alerting the developers when a "small tweak" actually has a significant impact on the broader user experience. As the release approaches, it might mean helping the product owner understand what percentage of the application's risks are tested and passing.

Yet, most teams focus on measuring non-actionable "counting" metrics, such as number of tests, that are hardly the right feedback for any stakeholder—at any time.

Scale

Most test automation initiatives start with the highest-performing teams in the organization. It makes sense. They're typically the most eager to take on new challenges—and the best prepared to drive the new project to success. Chances are that if you look at any organization with pockets of test automation success, you will find that it's achieved by their elite teams.

This is a great start...but it must scale throughout the entire organization to achieve the speed, accuracy, and visibility required for today's accelerated, highly-automated software delivery processes.

BREAKING THROUGH ENTERPRISE TEST AUTOMATION BARRIERS

How can mature companies with complex systems achieve the level of test automation that modern delivery schedules and processes demand? There are four strategies that have helped many organizations finally break through the test automation barrier: Simplify automation across the technology stack, end the test maintenance nightmare, shift to API testing, and choose the right tools for your needs.

If you'd like to explore these and other enterprise test automation challenges in-depth, I invite you to read my recent book, Enterprise Continuous Testing: Transforming Testing for Agile and DevOps.

DISCLAIMER: Note, the information provided in this statement should not be considered as legal advice. Readers are cautioned not to place undue reliance on these statements, and they should not be relied upon in making purchasing decisions or for achieving compliance to legal regulations.

ABOUT TRICENTIS

Tricentis is the global leader in enterprise continuous testing, widely credited for reinventing software testing and delivery for DevOps and agile environments. The Tricentis AI-based, continuous testing platform provides automated testing and real-time business risk insight across your DevOps pipeline. This enables enterprises to accelerate their digital transformation by dramatically increasing software release speed, reducing costs, and improving software quality. Tricentis has been widely recognized as the leader by all major industry analysts, including being named the leader in Gartner's Magic Quadrant five years in a row. Tricentis has more than 1,800 customers, including the largest brands in the world, such as Accenture, Coca-Cola, Nationwide Insurance, Allianz, Telstra, Dolby, RBS, and Zappos.

To learn more, visit www.tricentis.com or follow us on LinkedIn, Twitter, and Facebook.

AMERICAS

2570 W El Camino Real,
Suite 540
Mountain View, CA 94040
Unites States of America
office@tricentis.com
+1-650-383-8329

EMEA

Leonard-Bernstein-Straße 10
1220 Vienna
Austria
office@tricentis.com
+43 1 263 24 09 – 0

APAC

2-12 Foveaux Street
Surry Hills NSW 2010,
Australia
frontdesk.apac@tricentis.com
+61 2 8458 0766