Overcoming the Test Automation Gap

The Essential Guide to Transforming Your Automation Goals into Reality

Table of Contents

The Test Automation Paradox

Test automation is hardly a new technology; many of the frameworks and tools we use today first emerged on the market over 20 years ago. Yet, nearly everyone who uses automated testing still struggles with the test automation gap.

Simply put, the test automation gap is the difference between the number of tests you'd like to automate and the reality of your automation suite, and that gap doesn't appear to be closing for most teams.

In our annual Software Testing and Quality Report, we survey thousands of QA and DevOps teams each year to assess and compare various benchmarks, including the test automation gap. QA teams consistently fall short of their automation goals. While teams set increasingly ambitious targets each year—aiming to automate 63% of tests by the end of 2025, up from a 54% target in 2020—actual automation rates have remained flat at around 40% for the past five years.

QA teams have more advanced tools and efficient methodologies than ever before, so why is the test automation gap only growing wider, costing teams not just time and test coverage, but also hindering their ability to scale and keep up with modern demands? That's the question we're here to answer.

Although it may be a comfort to know you're not struggling alone, just because the test automation gap is a widespread problem doesn't mean it's not a critical one. Consistently falling behind on your automation goals directly impacts software quality, release speed, and team efficiency. As the demand for faster, higher-quality releases continues to rise, closing the test automation gap is more crucial than ever.

This ebook is designed for QA engineers, test automation engineers, QA managers, software development leaders, and software developers who are looking for actionable insights and strategies to help accelerate their automation efforts. Your test automation gap may feel like an impassable chasm. We'll help you pinpoint the root causes and build a strategy to bridge the divide.

Section 1

The State of Test Automation Today

The State of Test Automation Today

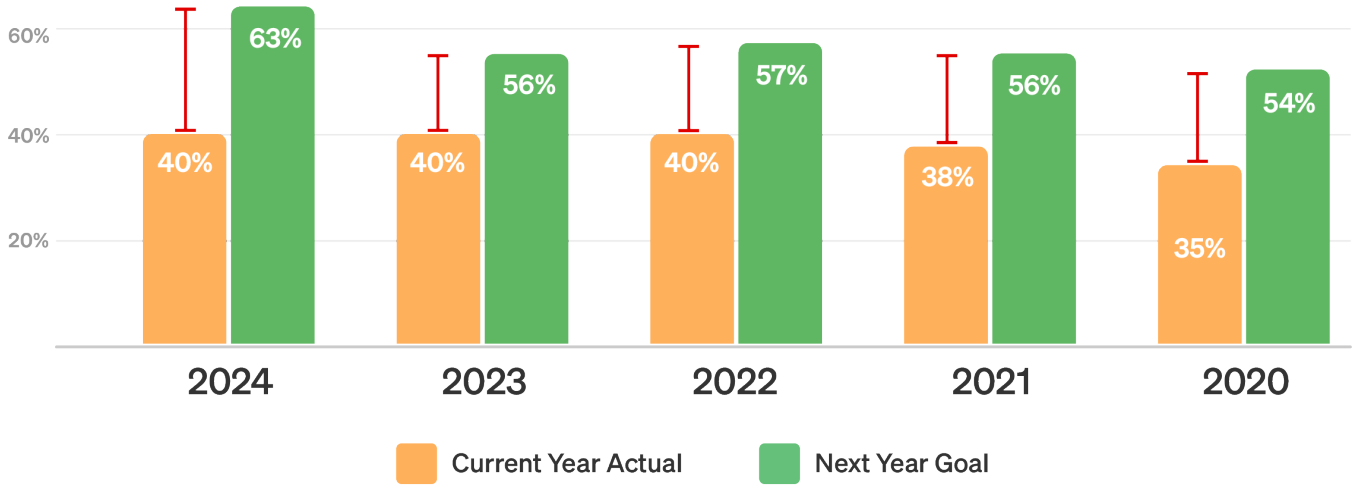
As mentioned in the introduction, TestRail publishes an annual report on the state of the QA industry, titled the Software Testing and Quality Report. This report delivers statistics, benchmarks, and insights into testing tools, teams, practices, and priorities—all based on feedback from thousands of QA professionals like yourself.

Test automation was a hot topic for the Software Testing and Quality Report this year, as it has been every year since its inception. Automation seems to be a recurring theme in both wins and challenges, a source of frustration in the present but one that holds optimism for the future.

To help you benchmark your own QA team against your industry peers, here is a selection of findings from the Fourth Edition of the Software Testing and Quality Report.

Current Automation Rates vs. Expected Automation Rates

What percentage of tests do you expect to be automated vs. manual in the next year?



Nothing illustrates the test automation gap better than this year-over-year expected versus the actual automation rate chart.

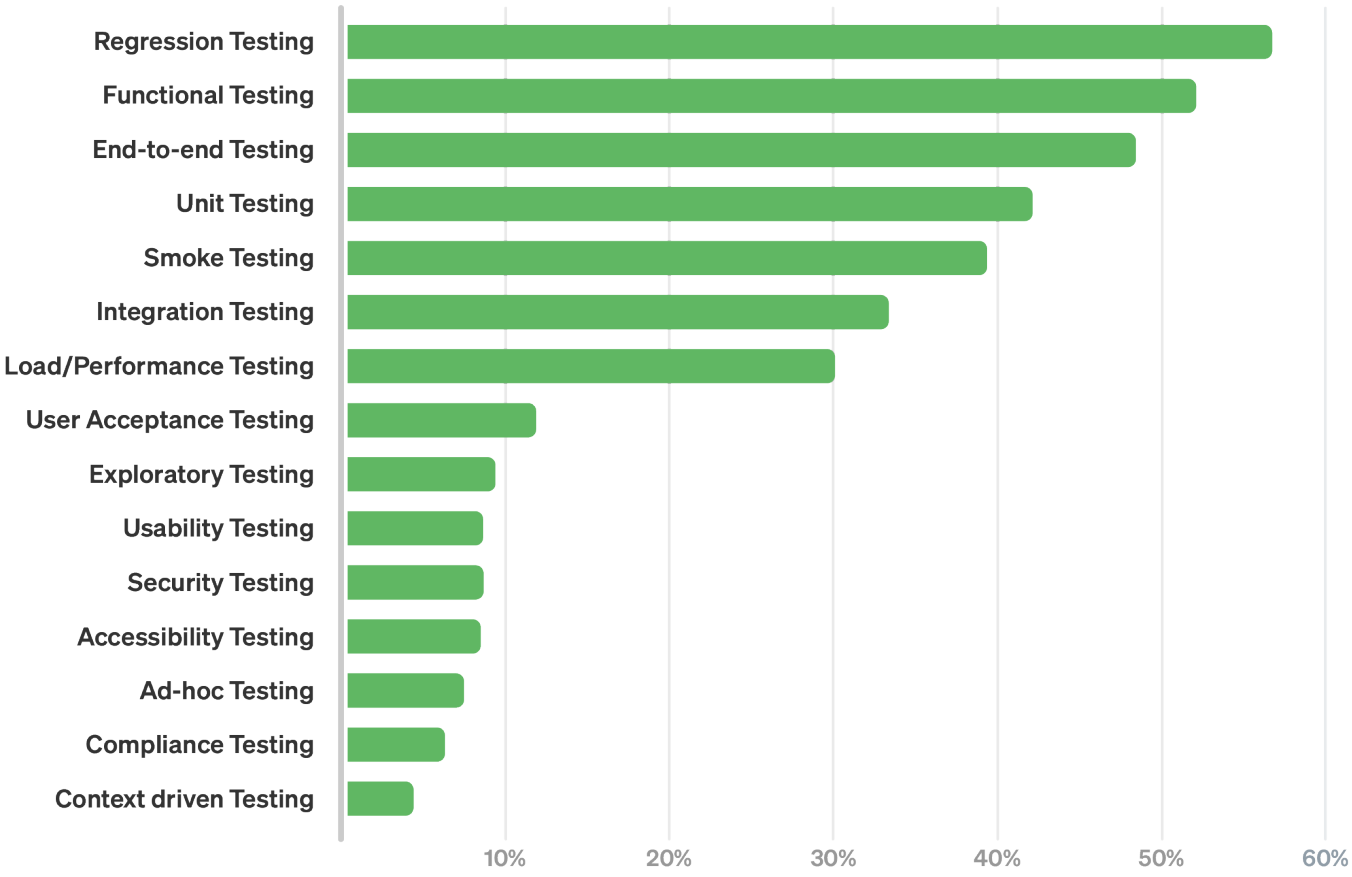
“Next year’s goal” has steadily risen each year, with a significant jump looking into 2025-perhaps due to the promise of AI, or perhaps just in response to ever-mounting pressure to release faster with fewer bugs.

Meanwhile, “current year actual” rates have risen much more slowly and seem stagnant at 40%. This suggests that, regardless of the effort teams put into increasing their automation output, they can’t keep up with the pace of demand.

These results show the test automation gap isn’t new, and it’s not improving for most teams.

Automated Testing Types

What kind of testing does your team run with automation?



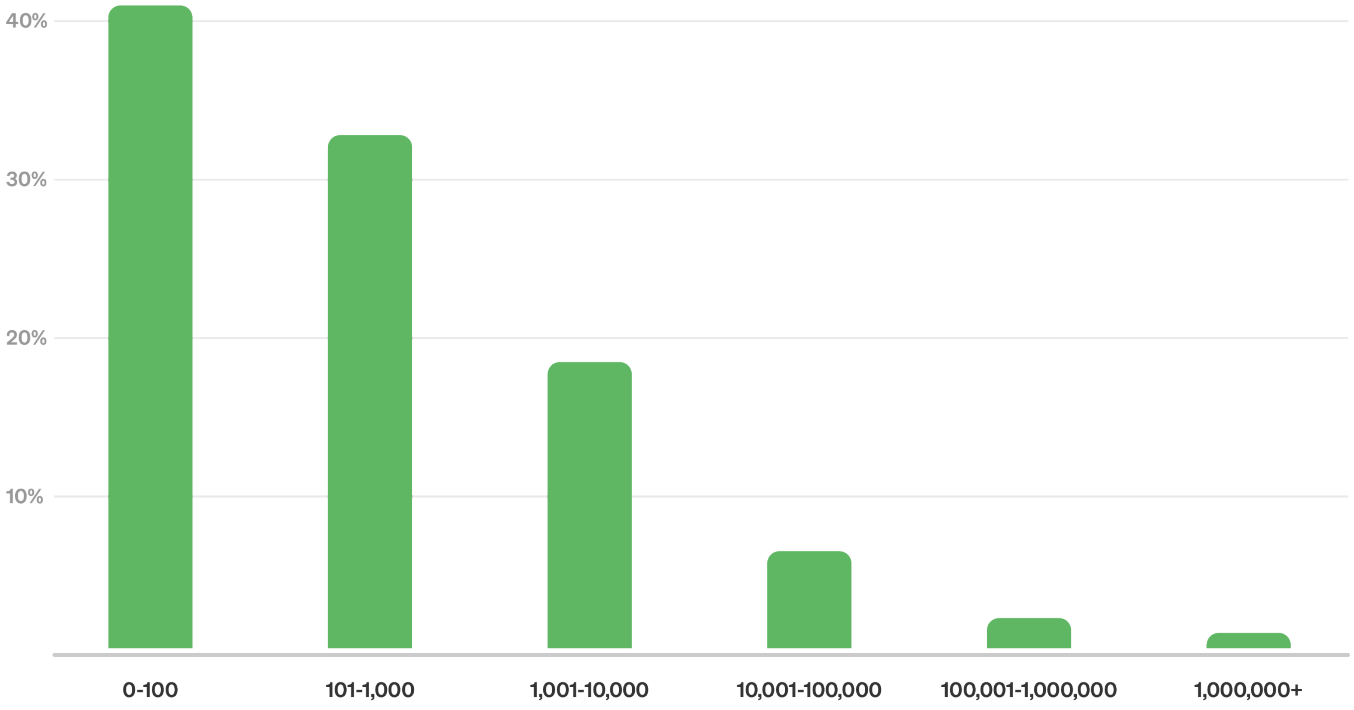
Survey responses show that most teams focus their automation efforts on test types that deliver high impact and are easy to repeat, such as regression and functional testing. These testing types can be time-consuming for human testers to conduct manually, so automating them speeds up release cycles, freeing testers to focus on tasks that require human insight.

Many of those tasks are other types of testing. It might sound surprising, but 100% automation is not and should never be the goal. Certain tests, like exploratory, usability, accessibility, security, and compliance testing, require human intuition and insights to be effective. Not only do such tests depend on human instinct, perception, and creativity, but attempting to automate them often creates more manual work than it resolves.

An efficient test automation program focuses on automating the right tests, not every test.

Average Number of Daily Automated Tests

On average, how many tests does your organization run each day?



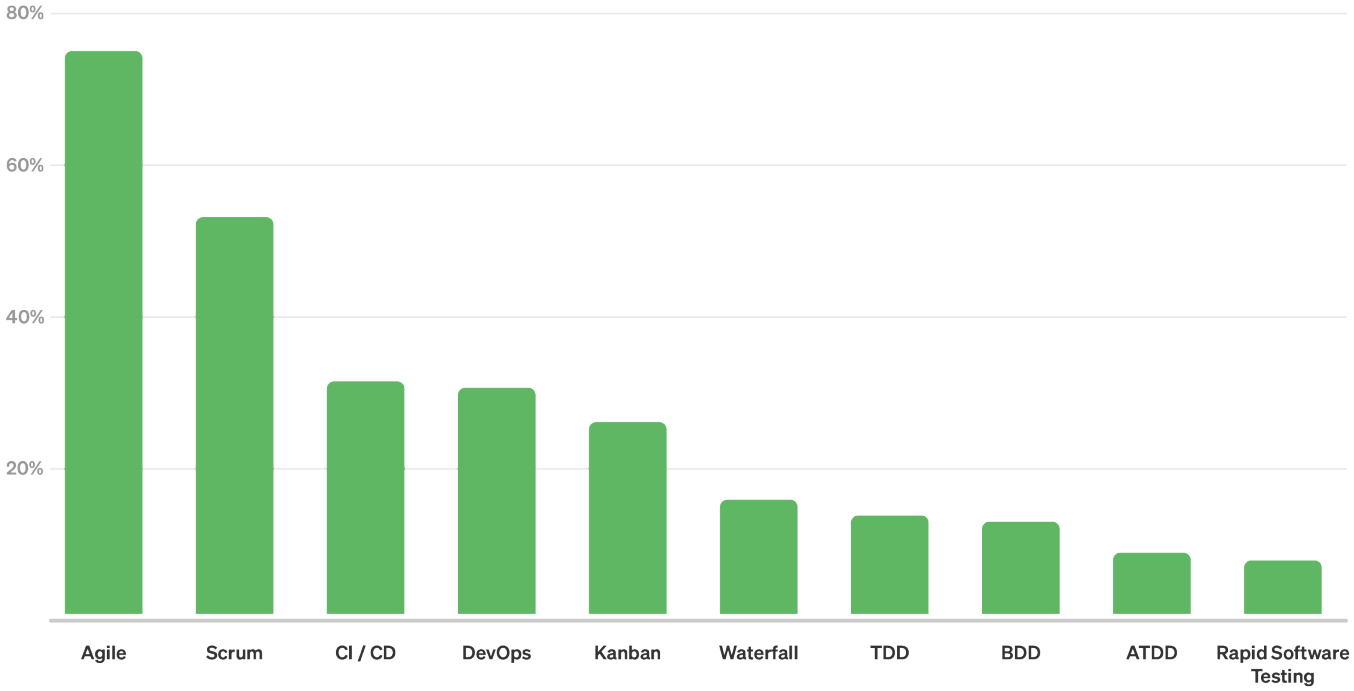
This chart illustrates the test automation maturity gap, with survey respondents reporting daily automated test runs ranging from zero to over one million. This reflects an enormous range of test automation maturity and infrastructure among modern QA teams.

Most teams are still in the early-to-mid stage of that range. About 41% of teams run fewer than 100 tests daily, while another 33% run between 101 and 1,000. This indicates that many QA teams are building their automation capabilities, but haven't yet reached full scale.

However, the right-hand side of this chart also shows what's possible at full scale with the right strategy and investments.

Continuous Integration/Continuous Delivery (CI/CD) Adoption

Does your team use any of the following development methodologies or techniques?



CI/CD is a critical piece of the test automation puzzle. Integrating test automation into your CI/CD pipeline triggers automated tests with every code change, enabling faster and higher-quality deployment while saving both tester and developer time.

About a third of teams reported using CI/CD, which shows that it has a sizable hold on the industry but still has plenty of room to grow. And for those who are utilizing CI/CD, the benefits reported by survey respondents were immense

- **86% of teams with high test automation and CI/CD integration report faster release cycles** compared to only 42% with low automation.
- **71% of teams with high test automation and CI/CD integration experience reduced defect leakage**, compared to 35% of low-automation teams.
- **58% of teams using automated test execution tied to CI/CD report a measurable return on investment (ROI)** in automation within 6 months.

Establishing a CI/CD pipeline may not be the best fit for every QA team, shape, and size, but for those with the resources and testing volume to justify one, the benefits speak for themselves.

Where Do We All Hope to Be Next Year?

In addition to the ever-elusive goal of "automating more tests," teams hope to improve on a variety of everyday automation struggles.

- **45% Want Fewer Test Breakages**
  UI changes and frequent application updates often cause automated tests to fail, leading to high maintenance costs and reduced reliability.
- **32% Want Better Test Data Management**
  Creating and maintaining test data is time-consuming, and many teams lack streamlined processes to support fast, consistent automation.
- **30% Want to Hire More Skilled Personnel**
  A shortage of experienced automation engineers makes it harder for teams to build and maintain reliable test suites.
- **29% Need Help Selecting Better Tools**
  With so many tools on the market, teams struggle to find one that fits their specific needs, tech stack, and application complexity.
- **26% Want to Reduce Flaky and Unreliable Tests**
  Inconsistent test results slow teams down and make it harder to trust automation as part of the CI/CD pipeline.
- **26% Want Better Test Coverage**
  Many teams say they still can't automate enough of their testing, leaving gaps in their quality assurance strategy.
- **21% Want Faster Test Execution**
  Long test runs can create bottlenecks in release workflows, making it harder to maintain fast feedback cycles.
- **19% Want More Robust CI/CD Integration**
  Some organizations are still working on better connecting their automation with DevOps tools and workflows.

Additionally, teams see considerable promise in AI for test automation. 29% of survey respondents are already seeing increased efficiency and automation in testing thanks to AI, with "AI-driven test automation and self-healing test scripts" being the most highly anticipated AI opportunity in QA over the next five years.

Teams seem aware of their automation shortcomings and are investing time and resources to improve. So why does the gap between test automation goals and reality persist year after year?

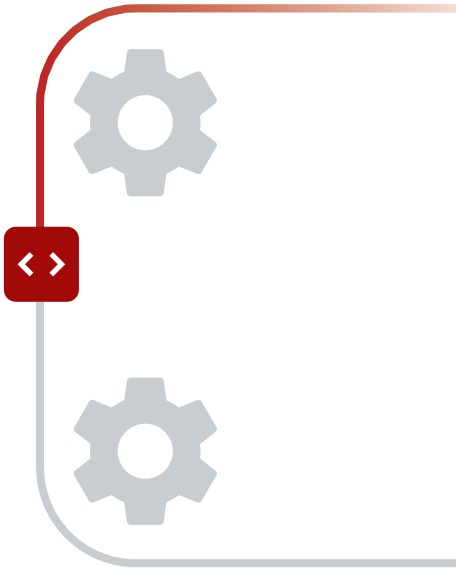
Section 2

Why QA Teams Struggle to Achieve Automation Goals

Despite growing pressure to automate testing, many QA teams face persistent roadblocks that prevent them from reaching their automation goals. These challenges are often deeply rooted in team capabilities, technical complexities, and organizational dynamics.

Skill & Expertise Gaps

A significant hurdle is the testers' lack of coding experience, which makes it challenging to build or manage automation frameworks effectively.



Hiring skilled automation engineers is also challenging, as demand outpaces supply. Even when companies invest in upskilling their existing QA staff, balancing learning new skills with day-to-day responsibilities is tough.

This contributes to a broader talent gap in the QA space, especially for roles requiring testing and programming expertise. Furthermore, many teams simply don't know how to design and implement a robust test automation strategy, leaving them without a clear direction.

Applications Under Test (AUT) Complexity

Another key factor is the growing complexity of modern applications. Distributed systems with cloud infrastructure, multi-tenant architectures, and microservices make testing far more intricate.

Many existing test tools struggle to keep up with these environments, lacking the flexibility or integration capabilities required to support comprehensive automation in such contexts.

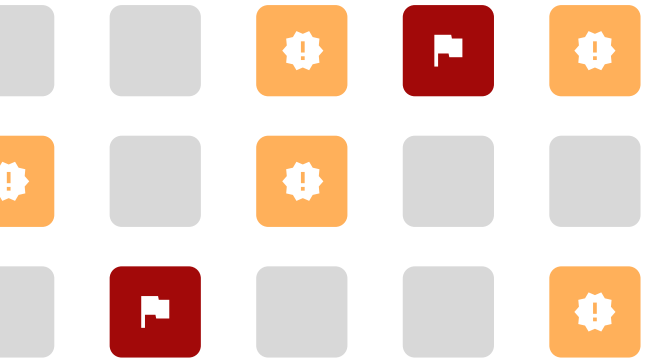
Tool Fragmentation & Lack of Centralized Management

Disjointed tooling across teams often results in fragmented automation efforts. Different groups may use their own tools and processes, which leads to siloed automation that's hard to scale or manage centrally.

Integrating test data from both manual and automated sources becomes difficult, hindering practical analysis and reporting. Without a centralized system to manage and track test execution and results, teams struggle with inefficiencies and blind spots across their testing landscape.

Time & Resource Constraints

Even when teams are committed to automation, time and resources are often limited. Automating complex scenarios can be labor-intensive, and in fast-paced release cycles, it's rarely prioritized over immediate deliverables. Few teams have the bandwidth or dedicated resources to regularly maintain and update automation scripts, which stalls long-term progress and leads to degraded test suites.



Test Stability & Flakiness Issues

Unstable automated tests are a common pain point. High maintenance costs and frequent test failures, often caused by changes in the UI or dynamically generated content, make it difficult for teams to rely on automation. Debugging these failures, especially when dealing with false positives or negatives, consumes valuable time and undermines confidence in the test suite.

Inadequate Test Coverage Strategies

Another common issue is an unbalanced approach to test coverage. Many teams overfocus on UI-level automation while neglecting other critical layers, such as API, integration, or unit testing. Without a structured approach to prioritizing what should be automated, efforts are often misaligned with actual risk and impact areas, resulting in a poor return on investment.

Lack of Executive Buy-In & Organizational Support

Finally, automation efforts often falter without strong executive support. Leadership may remain skeptical about the return on investment, especially when benefits like time savings or increased reliability aren't demonstrated. Without the ability to quantify and communicate the value of automation, teams struggle to secure the resources and buy-in needed to scale their efforts.

Section 3

Accelerating Test Automation: Strategies for Success

If any of the common challenges described above feel familiar, you are not alone. The good news is that the test automation gap can be bridged, no matter how insurmountable it looks from your current viewpoint. Even small, incremental changes in one or more of the following areas can start paying quick dividends for your automation progress.

Promote Training & Upskilling

Training and certification programs are a way to earn valuable credentials that give your team structure and momentum. For newer testers, especially, proper training can turn a vast and confusing landscape into a precise rhythm: learn, try, reflect, repeat. Each completed module boosts not just skill, but identity.

Supporting and prioritizing training for your team conveys that learning is part of the job, not an afterthought. Over time, this changes team culture: knowledge sharing increases, conversations shift toward patterns and practices, and hiring becomes easier with clear growth paths. In this way, training becomes more than a stopgap for skill gaps; it's the infrastructure for continuous improvement.

Action Plan

- Identify and support relevant certification programs for your team.
- Encourage regular learning as part of weekly workflow (e.g., 1 hour/week).
- Create space for internal knowledge sharing (e.g., lightning talks, lunch and learns).
- Make skill development part of performance reviews and career paths.

Choose the Right Tool for the Job

There are many different automation tools on the market, and any one of them could be the best choice for your team. While review sites and testing influencers can provide guidance, only you can determine the best tool for your specific needs.

When considering automation tools, it's vital to asses your application's complexity, tech stack components, organizational automation maturity, team technical skills, and, of course, your budget.

For lower maturity teams who are just getting started with test automation and have straightforward test requirements, low-code/no-code solutions are likely to be a good fit. For mature teams that are deep into their automation journey and have complex test requirements, a more robust platform is likely needed. And for any solution, it has to integrate into your existing tech stack to complement your workflow.

Action Plan

- Assess your team makeup, current automation program, and tech stack.
- Research automation tool options and compare them against your needs and abilities.
- Use a trial for tools to ensure they work as anticipated and integrate with your workflow.
- Put an onboarding program in place to train your team on how to best use the tool.

Encourage Cross-Team Knowledge Sharing

Pairing manual testers with automation engineers is one of those low-lift strategies that yields significant results. Manual testers have intuition about the product, the user's intent, and the edge cases that don't appear in the documentation. Automation engineers bring fluency in tools, frameworks, and technical workflows. When these perspectives merge, you get better automation.

This turns knowledge into something shared and relational. The manual tester gains confidence with technical tools by seeing them in action and connecting theory to the work. The automation engineer, in turn, benefits from test ideas and exploratory thinking they might not have previously considered. This creates an environment where automated testing is thoughtful and strategic.

Action Plan

- Pair manual testers with automation engineers on a regular basis.
- Run shared test design sessions before major sprints.
- Document and circulate reusable patterns and lessons learned.
- Rotate team members across roles to build empathy and skill depth.

Use Automation & Test Management Tools

We often hear that test automation lives in multiple places, like scripts on someone's laptop, results in a CI dashboard, or documentation in a spreadsheet. All of that process fragmentation makes it hard to see the bigger picture. Centralizing automation through test management tools brings everything into focus by creating a single home for manual and automated testing. For testers, this means more clarity on what's working, what's covered, and what still needs attention.

When tools like Ranorex and TestRail are integrated, you can immediately feel the difference in day-to-day flow. An automated test created in Ranorex can be linked directly to a test case in TestRail, where it lives alongside manual steps, history, and test run context. You get real-time visibility into execution results and more meaningful collaboration.

Action Plan

- Centralize all test cases, manual and automated, into a unified platform.
- Integrate tools like Ranorex and TestRail for traceability and visibility.
- Link test cases to user stories and CI/CD events.
- Review coverage and execution data regularly with the team.

Build Reliable, Resilient Automated Tests

Accelerating test automation means nothing if your tests are flaky or brittle. Reliability should be a design goal from the start. Use explicit waits over arbitrary delays, isolate tests to avoid shared state issues, and run them in clean environments. Layer your test suite wisely: unit tests for speed and precision, API tests for logic validation, and UI tests only where end-to-end coverage matters most.

Resilience is equally critical, especially in UI automation. Interface changes and tests that break with every minor tweak create drag instead of lift. Focus on the intent behind interactions. Are you testing button placement or the user experience? The more your tests align with user behavior and system goals, the better they'll withstand shifting UIs and evolving codebases.

Additionally, your tool choice can help foster resilience. Consider selecting an automation tool with features and capabilities that lend themselves to building reliable and resilient tests, such as self-healing and object recognition. While such features can't fully automate test design, they can dramatically reduce flakiness, freeing up time that your human testers can focus elsewhere.

Action Plan

- Select a tool that fosters reliable test design, and design tests to run independently in clean environments.
- Use explicit waits and verify the system state before interactions.
- Rely on stable selectors (e.g., data-test-id) and common actions.
- Focus assertions on user outcomes, not just UI sequences.

Utilize AI to Improve Test Automation Reliability

AI won't replace your testers or instantly stabilize flaky tests—its real value lies in augmenting human effort, especially in repetitive or complex tasks. While AI shows a lot of promise, AI-enhanced QA tooling hasn't quite hit the point of widespread adoption yet. One area where the industry is already seeing reliable AI-assisted results, however, is in test design and optimization.

Tools like DesignWise can leverage AI algorithms to more efficiently design test cases, generating recommendations that reduce wasted effort while still ensuring coverage and prioritization of critical paths. AI can also help identify coverage gaps, automate parameterization, generate Gherkin test scripts, and filter excessive test output to highlight what matters.

Still, AI comes with caveats. Generic models or misapplied automation can introduce silent failures if teams trust results without understanding them. Used wisely, AI expands what's possible, so testers can focus on judgment, prioritization, and thoughtful test design.

Action Plan

- AI — Leverage AI to help design efficient test cases.
- Use AI tools to identify gaps in test coverage.
- Analyze flakiness patterns to prioritize test maintenance.
- Review all AI-driven changes—don't trust blindly.

Create a Sustainable Strategy & Secure Buy-in

A strong automation strategy starts with clarity: define what you want to achieve, how you'll measure success, and what it will take to get there. Set realistic goals based on past efforts, and weigh both the risks and expected ROI. Not every test should be automated. Use criteria like frequency, criticality, and complexity to guide decisions. Lean on the test automation pyramid: prioritize unit tests for speed and reliability, support them with API and integration tests, and use UI tests sparingly where end-to-end coverage is essential.

Just as important as the plan is securing support. Share your roadmap with stakeholders, highlight how automation aligns with business goals, and identify clear metrics to track impact. With leadership buy-in and a focused, adaptable strategy, you'll be in a much stronger position to scale automation sustainably.

Action Plan

- Use Mike Cohn's test automation pyramid as your guide.
- Define success criteria and track ROI from day one.
- Apply automation selectively using the test pyramid model.
- Align automation goals with product and business priorities.
- Present your roadmap clearly and involve stakeholders early.

Section 4

The Future of Test Automation: Predictions for the Coming Years

Test automation is undergoing a major evolution. With smarter tools, faster release cycles, and more strategic QA roles, automation is taking on deeper responsibilities. Over the next few years, expect test automation to handle more complexity, integrate more seamlessly with DevOps, and make shift-left testing the norm. To keep pace, QA teams must stay agile, adopt new technologies, and rethink how they contribute to software delivery.

AI-Driven Test Automation

AI is quickly becoming a game-changer in test automation. Tools with self-healing capabilities can fix broken test scripts automatically, reducing manual upkeep. Powered by machine learning, these tools can generate test cases, uncover coverage gaps, and predict where bugs are most likely to appear based on historical data. While still emerging, AI is expected to evolve into a trusted co-pilot, helping testers accelerate delivery without compromising quality.

Shift-Left Testing Adoption

"Shift-left" is no longer just a buzzword. It has become a standard practice in the SDLC. Teams can catch and fix issues faster by starting testing earlier, often in the design or coding phase. This early feedback loop reduces rework and keeps quality front and center. With automated tests baked into CI/CD pipelines, testing becomes a continuous process, not a final gate. As release cycles speed up, shift-left testing will be essential to maintaining agility and stability.

Tighter Integration Between DevOps & Test Automation

DevOps and test automation are converging into a unified, automated workflow. Tests run with every code commit, providing immediate feedback and actionable insights. This tight integration improves code quality, speeds up releases, and ensures testing keeps pace with development. It also enhances collaboration between developers, testers, and operations teams, creating a smoother, more transparent delivery cycle.

What's Next? Closing the Test Automation Gap

Despite the maturity of testing tools and methodologies, many QA teams fall short of their automation goals. The reasons are clear and consistent: limited time, skill gaps, flaky tests, fragmented tools, and a lack of organizational support. These challenges slow down progress, but they're not insurmountable.

As automation becomes more advanced and integrated, testers have a growing opportunity to influence the entire development process, from coding to deployment. Skills like scripting, data analysis, and working with AI-powered tools are becoming essential. At the same time, low-code and no-code platforms are expanding the number of people who can contribute to automation. QA roles are evolving; embracing that shift is how teams will thrive.

Having the right tools is key. Solutions like Ranorex and TestRail help QA teams close the test automation gap by scaling automation efficiently, reducing complexity, and improving collaboration across the development lifecycle. Request a trial or watch a demo of one or both to see their capabilities in action.

TestRail

Start a free trial

On-demand demo

Ranorex

Start a free trial

On-demand demo

About TestRail

Gurock Software was founded in 2004 and now has offices in Frankfurt, Dublin, Austin, and Houston. Our flagship test case management solution, TestRail, is used by more than 100,000 members of development and QA teams to build rock-solid software, including companies like Amazon, NASA, Adobe, Sony, PayPal, and Siemens.

TestRail is the only platform that empowers QA teams to build, connect, and optimize all their testing processes. TestRail centralizes manual and automated test management and gives you visibility into your entire quality operation so you can manage your team more flexibly and build repeatable, scalable workflows. And, with a unified platform that integrates with your DevOps pipelines, you can share testing timelines, data, and insights across your whole organization.

TestRail is a leader in the G2 Grid for Test Management and Software Testing, with top ratings year-over-year for best results, most implementable, and overall enterprise leader. For more independently verified research and reviews, visit the TestRail page at G2 or Capterra.

Let TestRail lift your team out of chaos and toward faster, frictionless releases.

About Ranorex

Since 2007, Ranorex has empowered software teams with comprehensive UI testing tools that handle even the most difficult-to-automate interfaces.

Our flagship product, Ranorex Studio, provides all-in-one UI test automation across desktop, web, and mobile devices. Test automation experts can use Ranorex Studio's full IDE, which includes open APIs and tools for intelligent code completion, refactoring, debugging, and more. Automation novices can use Ranorex Studio's capture-and-replay tools and built-in methodology to rapidly build reliable, maintainable tests while expanding their automation skills. All members of cross-functional teams can collaborate on solutions by sharing reusable object repositories and test automation modules.

Ranorex tools help over 14,000 users worldwide deliver high-quality desktop, mobile, and web applications—read their success stories here. For more independently verified research and reviews, visit the Ranorex page at G2 or Capterra.