SMARTBEAR

TestComplete | White Paper

6 Ways to Measure the ROI of Automated Testing

The growth of new technologies and demand for faster release cycles are driving the standard for quality software through the roof. Trends like automation, continuous testing, and DevOps have raised the bar by introducing speed and flexibility into the software development lifecycle (SDLC), and everyone is scrambling to keep up. To stay competitive, teams today are pushed to optimize their testing and development processes to get more done in less time - all while keeping costs low. This perfect trifecta of speed, quality, and cost, is both the goal every team aspires to accomplish and the top challenge they face. Historically, quality assurance (QA) teams found they had to make tradeoffs between the three. Deliver faster, but risk bugs reaching production. Ensure quality, and gamble with keeping your release date. By implementing these new trends, especially automation, you won't have to choose.

As you look to adopt an automated testing process to meet the rising demand for faster delivery cycles and bug-free releases, it's vital to assess whether the return on investment (ROI) is worth the change. Before executing, or even thinking of building out an automation strategy, you'll want to calculate the net gain you'll see from transitioning. Divide this by the net investment needed to transition (i.e., the tools and resources you use), and you'll get your ROI for automated testing. This equation will look like the following:

$$\text{Automation ROI:} \quad \frac{\text{Gains} - \text{Investment}}{\text{Investment}}$$

The key question that arises is, *"what defines the gains and the investments?"* Calculate the 6 measurements outlined in this white paper to estimate the long and short term monetary value you will receive from investing in automation.

Before we dive into the benefits of transitioning and the investments in tools and resources you'll have to make, let's dive into the common pitfalls in calculating the ROI.

Common ROI Pitfalls

- **Only accounting for creating, developing, and maintaining automated tests versus manual tests.** Manual testing will always be important. While automation is on your mind, there are scenarios that will always require manually executed test cases. This means you'll still have to create, run, and maintain those tests.
- **Not accounting for the percentage of tests that need to stay manual.** Redundant or repetitive test steps are great candidates for automation, as having to run multiple of the same test type can be tedious and ultimately prone to human error. Test types that require human observation, such as determining whether a website is aesthetically pleasing or validating that the navigation menu is user-friendly, should remain manual. You'll find that these aren't easy to automate and won't provide a high ROI if you do.
- **Not syncing your automation tool stack with organizational capabilities.** To implement an automation strategy, you'll need to have both automation knowledge and product knowledge. This means you and your team need an understanding of both the application you're testing and the automation tools you plan to use.
- **Not accounting for test maintenance as an ongoing process.** After creating and implementing your automation strategy, you'll need to continuously maintain and update your tests. Over time, as you build new features and continue to make product improvements, your test cases and regression suites will grow. Ensuring these are usable over months or years will require continuous maintenance.
- **Not accounting for ROI over a period of time.** When building out a business case to transition to automation, you'll not only want to gauge the short-term benefits of investing, but also how it will impact your team and organization in the long run. Don't look at just one year, but determine what the three or five-year impact would be.

Not taking each of these pitfalls into account when calculating your ROI will skew your measurements, so it's essential for your success that you do. Transitioning to automation is a large investment and to effectively gauge the outcome, you will need a roadmap with accurate benchmarks. Every organization is different though. How you define, execute, and maintain your tests will vary. These variations will alter how you measure the ROI of automation. The benefits will be unique to your team.

Consider the following key variables that make your organization unique before getting started.

1. Business requirements to be implemented
2. Test cases for each requirement
3. Percentage of tests that can be automated
4. Test case complexity
5. Test cases from prior requirements needed for regression testing
6. The total number of configurations that need to be tested

While this is not an exhaustive list, it highlights the most common differentiators between teams and businesses within any industry.

Now that we've reviewed the pitfalls teams face when calculating the ROI of automated testing and the top six variables that differentiate your team and company from others, let's dive into the various metrics you need to calculate.

The Six Ways to Measure the ROI of Automated Testing

Start by breaking down the ROI equation into two parts and review how to calculate your gains as well as your investments.

$$\text{Automation ROI:} \quad \frac{\text{Gains} - \text{Investment}}{\text{Investment}}$$

The first step is to calculate the following six costs and cost savings to determine your ROI. We'll walk through the metrics you need for each and the best practices to keep in mind.

1. Automation of New Tests
2. Automation of Prior Tests
3. Coverage Across Environments
4. Defect Leakage
5. Test Redundancy & Reuse
6. Knowledge Leakage

1. Automation of New Tests

When starting to build an automation business case, the common first step is to look at what it will take to automate *new* tests cases. How much time will it take to develop, execute, and maintain an automated test? In this step, you will need to decide which tests can be automated and which ones should remain manual. To get the total cost, you'll also want to take the hourly cost of the number of team members executing the tests into account.

Calculation 1

The Cost of Automating New Tests

# of Total Test Cases	1,000
Hours to Develop, Execute and Maintain each Test	0.5
Total Automation Time	**500 Hrs**
Avg Cost of an Automation Engineer per Hour	$43
# of Automation Engineers	2
Total Cost	**$43K**

Best Practices

1. **Don't think of you ROI as automated vs. manual.** The key point to walk away with is to realize that some test cases should be manually executed. When deciding which ones to automate first, it's usually a good idea to start with the simpler and repetitive test steps. It's easy to fall into the trap of automating complex and time-consuming test cases first, but these will take longer to implement and lead to less returns. Transitioning the smaller tests out of the gate will enable you to start automating quicker, meaning so you can get more traction and see your ROI sooner.
2. **Factor in team members conducting both manual and automated tests.** Since you're going to run both manual and automated test cases, you'll need to consider how much time your team is spending creating and executing both.

2. Automation of Prior Tests

The second step in building an automation business case is to calculate the cost of automating your prior test cases, meaning your regression tests. Regression testing is the process of running older tests to ensure that new updates to a piece of software haven't introduced or re-introduced previously eradicated bugs. Regression testing is vital to your success as it will help you ensure that bugs stay dead and that product features that were already validated continue to function properly. Over time, these test suites will grow and will take longer to execute. Automation here will allow you to run regression tests quicker and boost your confidence in the next release.

Calculation 2

The Cost of Automating Regression Tests

# of Regression Tests	1,000
Hours to Maintain each Test	0.25
Total Automation Time	**250 Hrs**
Avg Cost of an Automation Engineer per Hour	$43
# of Automation Engineers	2
Total Cost	**$21.5K**

Best Practices

1. **Immediately integrate new automated tests with your existing regression testing suite.** When calculating your ROI, you should assume each new test will eventually become a regression test. Tie these into your regression testing strategy from the beginning and ensure you have a process in place to do so.
2. **Automate repetitive test cases with medium to high complexity.** You'll receive the most value for your efforts.
3. **Remember to account for the maintenance and development of new test cases as an ongoing process.** Unfortunately, no one works in a world where spending ten hours automating a set of tests means you won't have to deal with them again. Any test will require maintenance, so accounting for the time spent on each of these will give you the most accurate year-over-year view of your ROI.

3. Coverage Across Environments

The goal of automated testing is to improve software quality while testing faster and reducing costs, and there is more to the ROI of automation than accounting for manual and regression tests. The explosion of devices, browsers, and operating systems in the industry has expanded the number of environments, and combinations thereof, that you can run your tests on.

Unless you're certain your end users will only be able to access your software from a single environment, chances are you're developing a product that will need to be accessible across a variety of platforms. When calculating the costs of environment coverage, you'll want to consider whether or not you're running tests in parallel and if you require a device lab to properly ensure test coverage.

Without proper parallel testing and the coverage it can provide, you risk encountering defects further downstream. The sooner you catch a defect, the cheaper it is to fix it. The cost of fixing bugs later in your development cycle is the second metric you will use to calculate your total environment coverage.

Calculation 3

The Cost Savings from Ensuring Proper Test Coverage

Cost of Lab

- # of devices
- Average cost per device
- Cost for lab maintenance

Defect Leakage

- Defects that could go downstream due to lack of environment coverage
- Hours to fix downstream defects

Tool

- Cost of a Cross Browsing Tool

Best Practices

Increase environment coverage by running cross-browser tests, unit tests, regression tests and smoke tests in parallel.

1. **Cross-Browser Tests:** Testing across different browsers and devices is one of the most time-consuming aspects of validating the front-end of your website or web application. Running these tests in parallel will enable you to run hundreds, if not thousands, of tests across a wider variety of browsers and browser configurations.
2. **Regression Tests:** Deployments are happening at such a rapid pace and regression testing is one of the best ways to have a type of "testing version control." Regression testing will help you ensure the functionality of every new build matches that of the last stable build. Running these tests in parallel enables you to increase test coverage.
3. **Unit Tests:** According to the testing pyramid, unit tests should be the most common test type in your entire testing suite. Due to their abundance, often tens of thousands of tests, running these under an hour is only possible with a massive parallel testing infrastructure investment.
4. **Smoke Tests:** Need to get your minimum testing done in the next 20 minutes while you push a hot fix? The best way to get the most testing done in the shortest amount of time is to run smoke tests in parallel.

4. Reduction of Defect Leakage

Defect leakage refers to the amount of bugs or issues that end up in production due to not being found earlier in the software development lifecycle. This can be the result of numerous issues, such as poor environment or functional testing coverage. The key metrics you want to track are the costs (i.e., money and time) associated with having to address defect leakage post deployment.

Calculation 4

The Costs Associated with Defect Leakage

Defect Leakage

- From lack of functionality coverage
- From missing requirements
- Hours to fix downstream defects

Best Practices

Catch defects earlier in the SDLC. The sooner you test, the sooner you can start uncovering bugs. This is the core idea behind the growing trend of 'shifting left.' As opposed to the traditional waterfall model where testers and developers were clearly separated, 'shifting left' seeks to close the gap. Testers are tasked with validating an update to a piece of software at the time of creation. Developers could even be running tests while they're still in the development phase. The end goal here is to reduce costs and by catching bugs sooner, you can save an impressive amount of money and time.

5. Test Redundancy and Reusability

The purpose of tracking the reusability of your tests and the redundant steps you're taking is to avoid duplicating testing efforts. Why recreate the wheel for tests you already built when you can reuse them? By reusing tests, you'll be able to improve your testing cycles and ultimately, improve test speed. To calculate this cost, you'll want to know the number of tests that were recorded more than once or the number of tests that have duplicate components, the amount of time spent searching for redundant test cases, and the time it takes you to develop and execute those redundant tests.

Calculation 5

The Costs Savings from Reducing Redundant Tests

- No. of tests that were recorded twice or have similar components
- Time spent searching for redundant test cases
- Time to develop and execute redundant test

Best Practices

1. **Build modular test scripts to enable test reusability.** Take a step back and look at your entire testing framework to determine which tests are being run the most often (i.e., which test are you repeating that take a long time). Take a typical 'logging in' action as an example. To run an effective logging in test, there is usually a ridiculously high number of variations you need to run through. Why would you not want to

create a test that can be reused to test every variation of a log in?

2. **Leverage a test case management tool that can help you search for duplicate test scripts.** Tools such as QAComplete will enable you to store tests with custom fields so you can personalize them to your organization or industry. Enabling yourself to do this will help you search for those redundant test cases in a much simpler and quicker fashion.

6. Reduction of Knowledge Leakage

This is often the most forgotten measurement when calculating automation ROI. The average tenure for a test engineer at any given company is typically three to five years. When they leave, there is always some degree of institutional knowledge that is lost. When building out a long-term ROI case for moving to automation, it's essential you take this into account. There will be test cases that need to be re-engineered after an individual is gone, and this can be costly.

Calculation 6

The Cost Savings from Reducing Knowledge Leakage

- Average tenure for a test engineer or have similar components
- Time to re-engineer lost cases

Best Practices

1. **Process documentation.** The best way to prevent losing as much knowledge as possible is to document your test process. Having good test documentation will enable the next engineer to re-build lost cases quickly. It will also allow you to reassess whether or not those test cases are still important.
2. **Test case management.** Leverage custom fields in test case properties to personalize search for your organization.

Automation Investments: Tools & Resources

Next, let's review the second half of the equation – the investments in tools and resources needed to transition to an automated testing process.

Automation ROI:

$$\frac{\text{Gains} - \text{Investment}}{\text{Investment}}$$

1. *Tools*

   There are any number of automated testing tools available on the market today, and choosing the one that is right for you can be a daunting task. When looking at a tool, you not only want to assess its functionality and costs, but also the time-related resources needed to implement it effectively and efficiently. How long will it take your team to fully ramp up and be comfortable executing tests with the product? Is it an easy-to-use tool for everyone, or will some team members need additional training? Some automated testing tools rely heavily on scripting tests while others will enable less technical testers to create scripts through point-and-click actions. You'll also want to consider if the vendor you're buying from provides enough resources to support you. Do you even have dedicated support? Is there sufficient documentation, trainings, or even white papers and webinars that you can leverage to educate and train your team?

2. *Resources*

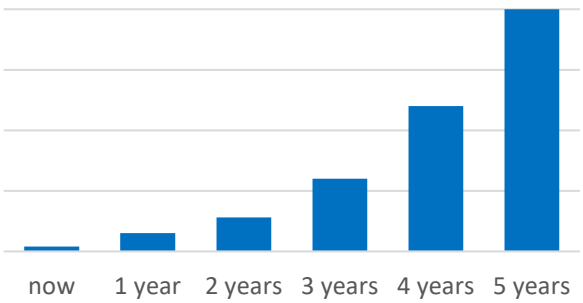
   It's rare that everyone on your team will have automation experience, so when it comes to resources, you're really gauging how much time it will take for your tool to make an impact. You'll either need to invest in growing your team's automation knowledge organically, via the resources mentioned above, or inorganically, through hiring.

Your Automated Testing ROI

Now that we've discussed what metrics you need to calculate your gains from automated testing and how much you'll spend to get there, you can build out your ROI for both long and short-term. The results will look like the following:

Our team can save **[INPUT 1st Year $$]** after one year. Over **[INPUT ROI PERIOD]**, we can save **[INPUT ROI $$]**.

You'll find that over the course of one, two, or even five years, your monetary ROI from investing in automation will grow exponentially.



Alongside the quantitative value you will realize from your investment in transitioning, there are a few other qualitative benefits you will notice as well. For example, over the course of your first few years, your testing process and the results you see will mature and you'll maximize your application's test coverage. By continuously optimizing your testing suite, you'll enable your team to execute tests across environments faster and reduce the risks associated with redundancy and defect leakage. As the number of devices and technological platforms available to end users continues to grow, this will become essential to your success. With a solidified test process in place, you can spend more time focusing on building new features to enhance product quality – ultimately giving yourself the opportunity to stay competitive in the market.

TestComplete

Functional Test Automation for
Desktop, Mobile and Web

Get Started

About SmartBear Software

Supporting more than five million software professionals and over 20,000 companies in 194 countries, SmartBear is the leader in software quality tools for teams. The company's products help deliver the highest quality and best performing software possible while helping teams ship code at nearly impossible velocities. With products for API testing, UI testing, code review and performance monitoring across mobile, web and desktop applications, SmartBear equips every development, testing and operations team member with the tools to ensure quality at every stage of the software cycle. For more information, visit: http://www.smartbear.com, or for the SmartBear community, go to: Facebook, Twitter, LinkedIn.