Moving From Manual to Automated Testing

A Tester's Journey

CrossBrowserTesting

TABLE OF CONTENTS

Introduction 3

Why the Shift From Manual to Automation is Happening 4

Where to Begin 8

Things to Keep in Mind 11

Summary 11

Introduction

As developers and product teams strive to push new code and features out to customers faster than ever, many are realizing the advantages of automation. As they write new code, developers are slowly starting to include writing automate tests as a matter of the norm, instead of an extra tedious task. However, this transition is often easier said than done and many product teams struggle with it.

In this ebook, CrossBrowserTesting will explore why software teams have come to favor automated testing over manual. Moreover, we hope to serve as a guide for when the time comes for your organization to make the switch.

Why the Shift From Manual to Automation Is Happening

Start With Manual Testing

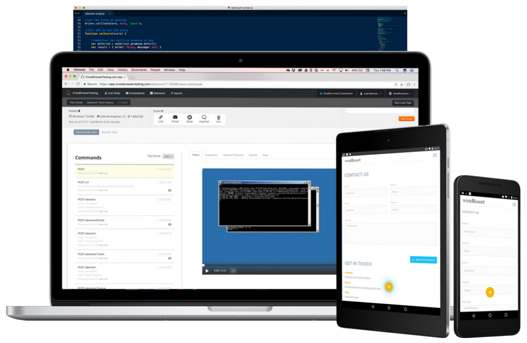
Manual testing is done completely by human hand, using the same tools that the end-user would operate: a computer mouse, keyboard, or the touch of their for mobile devices. This is simply because the manual tester acts as the end-user to explore unit, integration, and functional tests of the application. This also means that manual testers require no coding knowledge to perform their job functions.

Uses of Manual Testing

Manual testing will always be a critical practice to find and fix bugs for software. It is valuable for manual testers to act as the consumer's advocate in order to assess quality, responsiveness, and to an extent, user experience. Especially for exploratory testing of new features, there is no method that can replace a human's curiosity indetermining where bugs might lie in wait.

"However, this process of Continuous Integration and Delivery always demands continuous testing."

Tweet this Quote!



The other great part about manual testing, is that it can be done by almost any member of a software team or business. Because the UI of many products in early stages tends to change often, it may be beneficial to have team members manually test features until stabilization occurs. This will keep costs down in writing and maintaining automated tests.

If you are still finding large amounts of new bugs and defects when manually testing the feature or application, this process is probably **not yet ready for automated testing**. It is important to remember automated testing will only test what the script says, so the application should have a stable UI and a solid understanding of it's strengths and weaknesses.

The Rise of Automated Testing

Automated testing uses the assistance of tools, scripts and software to perform test cases by repeating predefined manual actions. The biggest draw of automated testing is that it's noticeably easier to run tests quickly, allowing for larger test coverage across environments, and larger code coverage during the testing phase.

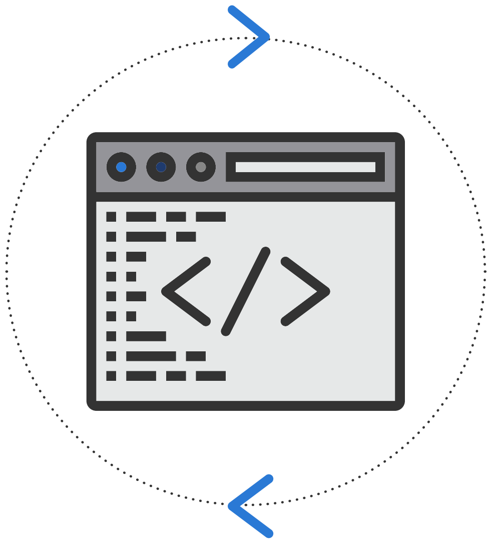
Agile vs Waterfall

One reason for the conversion from manual to automation is the recent shift from Waterfall to Agile development.

While Waterfall is a non-iterative and sequential design process, the newer, incremental approach, termed Agile, allows for more flexibility, feedback, and testing throughout the software development process.

This is because an Agile strategy caters to changing requirements throughout the development process by having team members integrate their work frequently and communicate more often with customers, business owners, and designers.

However, this process of Continuous Integration and Delivery always demands continuous testing. Manual testing no longer had the luxury of a 2 week testing window, but was now facing testing windows of just 2 hours. Successful Agile development teams implement a large amount of test automation throughout the delivery process to make 2 hour testing windows a reality.



The Introduction of Selenium

Automation has not always been the ideal option for testers. Most testing could be done manually since there were only a few devices and browser versions that need to be testing on, and most teams followed a Waterfall methodology.

Additionally, methods for automating tests used to be expensive and out-of-reach for software development teams, so they were never a cost-effective or pragmatic option for QA.

That all changed with the introduction of Selenium WebDriver, an open-source tool that allowed users to automate test scripts written in any language, and here's the real kicker, run across different browsers and operating systems.

The number of devices, browsers, and operating systems has rapidly grown since the beginning days of web development. **A process that was once straightforward for a manual tester has been complicated by the need to test across all of these environments**.

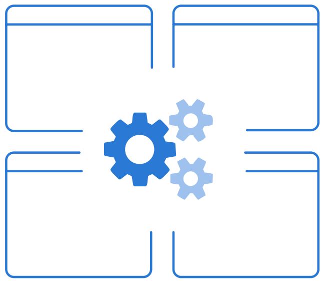
Developers and QA Engineers can take advantage of Selenium's robust WebDriver across a variety of these environments, and can even run them in parallel using Selenium grid or a cloud solution like CrossBrowserTesting.

Regression Testing

Regression testing has always been a way for testers to ensure quality software from release to release, but it is a cornerstone of agile testing methodologies and automated testing makes true regression testing possible.

Regression testing ensures that new code pushed to the testing server does not break any of the old code or application. Because of the Agile's iterative nature, development and testing teams can feel confident when deploying multiple times a day as long as their regression tests pass during each building phase.

"The need to start automating will become increasingly apparent to developers who are committing code often and testers who are working to ship quickly on a stable product."



Parallel Testing

Because automated testing is run by scripts on machines, the cost of additional environments to run them on becomes only a capital cost and not a labor cost. Multiple tests can be run at once by building a Selenium Grid or other testing grids in-house, which may involve VMs, Simulators, or real devices.

Unit and GUI tests can be ran concurrently on different environments, either speeding up a testing suite's execution time or improving test coverage.

Parallel testing's impact on software delivery has grown alongside the growth of the cloud. Cloud testing platforms, like CrossBrowserTesting, allow you to run tests in parallel against hundreds of machines at once, on real mobile devices and browsers.

Development teams can now go fully continuous, testing and deploying at speeds that allow for more innovation and ultimately happier customers.

When to Automate

The need to start automating will become increasingly apparent to developers who are committing code often and testers who are working to ship quickly on a stable product.

As stated, a team following Agile will realize the need to implement automation starting from the Unit layer, and working through to the GUI layer. More plainly, though, **automation truly become necessary when you need to start repeating tests more than once.**

It becomes tedious and time-consuming for testers to go back and manually test a web application that has an established UI on different devices and browsers every time a small change is made -- such as the color of a button in a unit test -- to make sure that change hasn't created new bugs in the rest of the code.

When you feel like you are repeating a test over and over on a daily, weekly, or even monthly basis it may be time to automate it. You will improve your overall testing ROI in time and money by investing in the upfront cost of automation.

"While collaboration between testing and development teams should always be a priority, it becomes even more essential as you move toward automation."

Where to Begin

Testers

The most difficult transition will likely be for testers who have been performing fully manual testing as they will need to be trained to acquire at least a basic knowledge of coding and automation concepts, as well as an understanding of the different automation tools and their purposes.

If the team is starting from scratch with no automation experts, this process will take some time, and you'll probably need to bring some experienced programmers on board including automation architects and engineers.

These additions will be an asset as thought leaders when it comes time to choose automation tools, prioritize which tests should be automated,build frameworks, as well as convert, create, and execute automated tests.

Meanwhile, manual testers can begin to focus on writing simple tests, executing tests, and evaluating results as they get up to speed with a framework like Selenium..

The other advantage to increasing test automation is that it concurrently frees up their time to focus on the projects that require manual testing to leverage their specific skills and precise observation.

Company-wide Initiatives

While collaboration between testing and development teams should always be a priority, it becomes even more essential as you move toward automation.

In automated testing and Agile CI, having these two teams work hand-in-hand to communicate on issues, changes, and objectives is more important than ever. **Having siloed teams simply will not produce satisfactory results.**

Starting by setting specific goals and guidelines for both roles as well as your team's definition of "done" will ensure that everyone is on the same page throughout the development lifecycle in order to reduce time to market while progressing consistent quality.

Your approach to automation will also depend on the size of your team and the size of your

"In fact, cross-browser testing is a significant part of test automation to make sure that users on every device, browser, and operating system are afforded the same high-quality experience with your web application."

company. Setting individual goals and tracking accomplishments when you begin automating will help you measure success and evaluate best practices for moving forward.

Additionally, if you do hire automation engineers and architects from the outside, they will have to work with manual testers to get to know the current testing process, products, and expected future projects. Similar to testing and development, manual and automated testers need to have a give-and-take in order to influence a successful transition.

Execution

Before you can begin automation, you need to decide which tests you want to automate and which environments you want to automate them on.

That is to say that, especially at the beginning of the transition, you will have to choose which tests are your priority. Consumer facing web applications, high traffic pages, regular regression tests and high risk browsers should be prioritized. Additionally, focusing first on fewer and smaller tests will make it easier to maintain them and find bugs.

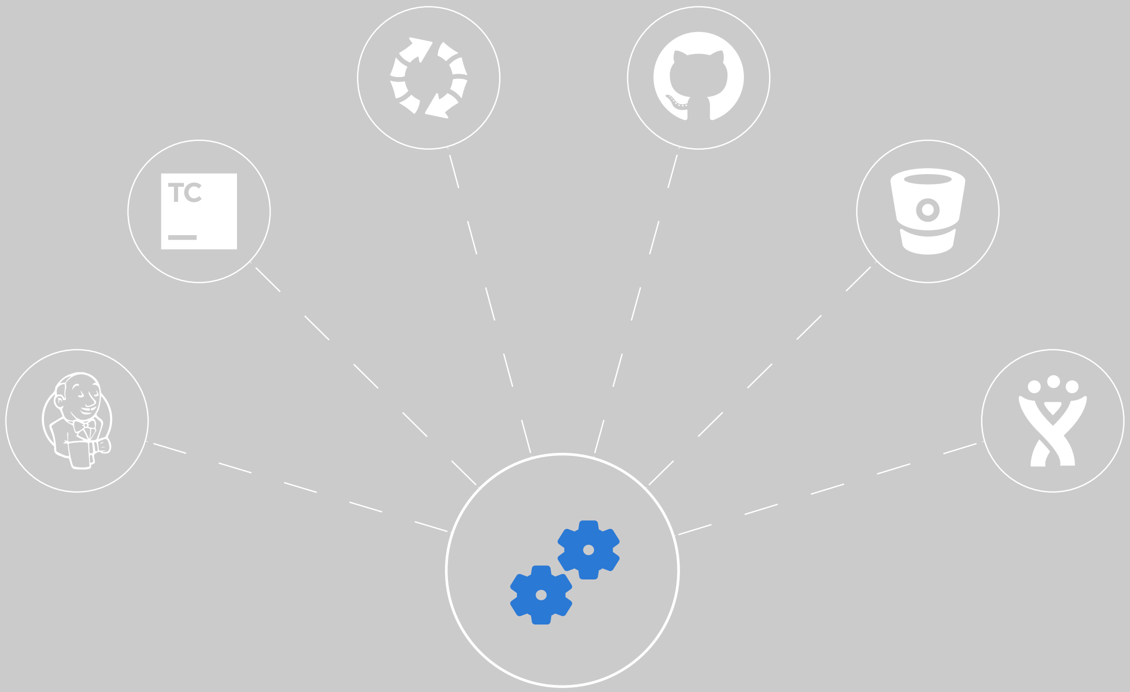
The other factor you have to consider is what environments you want to test on. It's not enough to automate tests on the latest Chrome browser on the newest Macbook with the most recent operating system that everyone in your organization uses.

In fact, cross-browser testing is a significant part of test automation to make sure that users on every device, browser, and operating system are afforded the same high-quality experience with your web application.

This means you will have to decide which combinations and configurations you want to test on to reach your market majority, as well as whether it's going to be executed through a Selenium Grid from a homemade device lab or through a third-party cloud platform.

Tools

Transitioning to automated testing will be a lot easier with the right tools complementing your team's efforts.



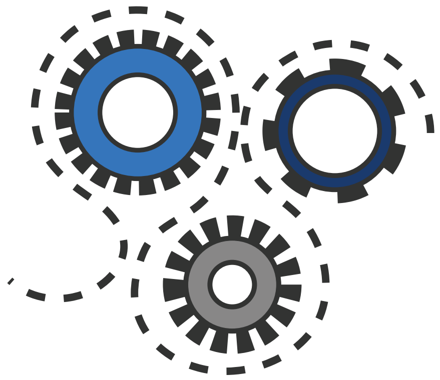
This decision will be based on your organization's technologies and projects and is where an automation architect's role will come in to consider details like what languages and environments different tools support.

The foundation of automated testing, as we mentioned, is Selenium WebDriver which allows you to run automated test scripts in different browsers. However, while Selenium is the most popular, there are other tools that can be used to automate tests such as TestComplete.

There are also many other open-source and paid accessories that you'll likely want to invest in a combination of to make automation easier, faster, and more enjoyable for testers.

Many test automation frameworks, for example, are free and much easier to implement than building one from scratch.

A framework is the rules, guidelines, and procedures that lay a groundwork to develop and execute automated tests.



The main categories for these automation frameworks include module, common library, data driven, keyword-driven, hybrid, and behavior-driven, and there are a diverse number of specific options to choose from.

Another element you may want to consider is paid, third-party tools for certain issues, which will create a noticeable difference in the quality of your web applications. For example, many teams will invest in paid tools for optimizing processes like performance testing, functional testing, screenshot comparisons, and cross-browser testing.

Things to Keep in Mind

Automation is not a silver bullet and is not going to automatically solve all your testing problems. Testers in automation have issues of their own such as false positives and flaky builds that come from a lack of human observation, maintenance, and company-wide collaboration.

In addition, though the ROI automation will likely be apparent over time, the transition to a mostly-automated process from a mostly-manual one will take commitment, patience, and financial investment. A complete work style change will not be a quick and easy one, and if it is, it will not be functional.

Furthermore, **you shouldn't expect to automate 100 percent of your tests**. Manual will always be a necessary and significant part of the development and testing process. Testing new projects, functions, and features manually before moving to automation is the best way to build the best application for clients, customers, and other users.

Summary

Augmenting manual with automation through parallel and regression tests on consistently updated or large projects is what will save you time, money, and frustration, while increasing effectiveness, efficiency, and coverage of tests. As the software industry continues to grow, test automation will become vital to development. Moreover, as your team explores and practices automation, its specific uses for your organization will become even more distinct over time.

The All-In-One Browser Testing Platform

One solution for all the testing needs of the modern Developer, Tester, and Designer.

Test Manually

Interactively use, test, and debug your app on remote browsers and devices, with access to native settings, inspection tools, and more.

Learn More

Take Screenshots

Capture full-page screenshots of your website across different browsers and devices at one time, then share with stakeholders.

Learn More

Automate It All

Run Selenium, Appium, and other open source testing frameworks against our collection of real devices and browser, right in the cloud.

Learn More

About CrossBrowserTesting

CrossBrowserTesting provides a cloud-based platform for Development and QA teams to run manual, automated, or visual tests on thousands of real browsers and mobile devices. We are a leader in Selenium and Appium testing solutions, providing one of the largest real device, cloud-based grids in the world.

Choosing our platform for your browser testing means you'll see higher quality software releases at a faster pace. Imagine being able to run your 2 hour testing suite in just minutes across a wider range of browsers and devices? Well, we're making that a daily reality.

CrossBrowserTesting

A SMARTBEAR COMPANY