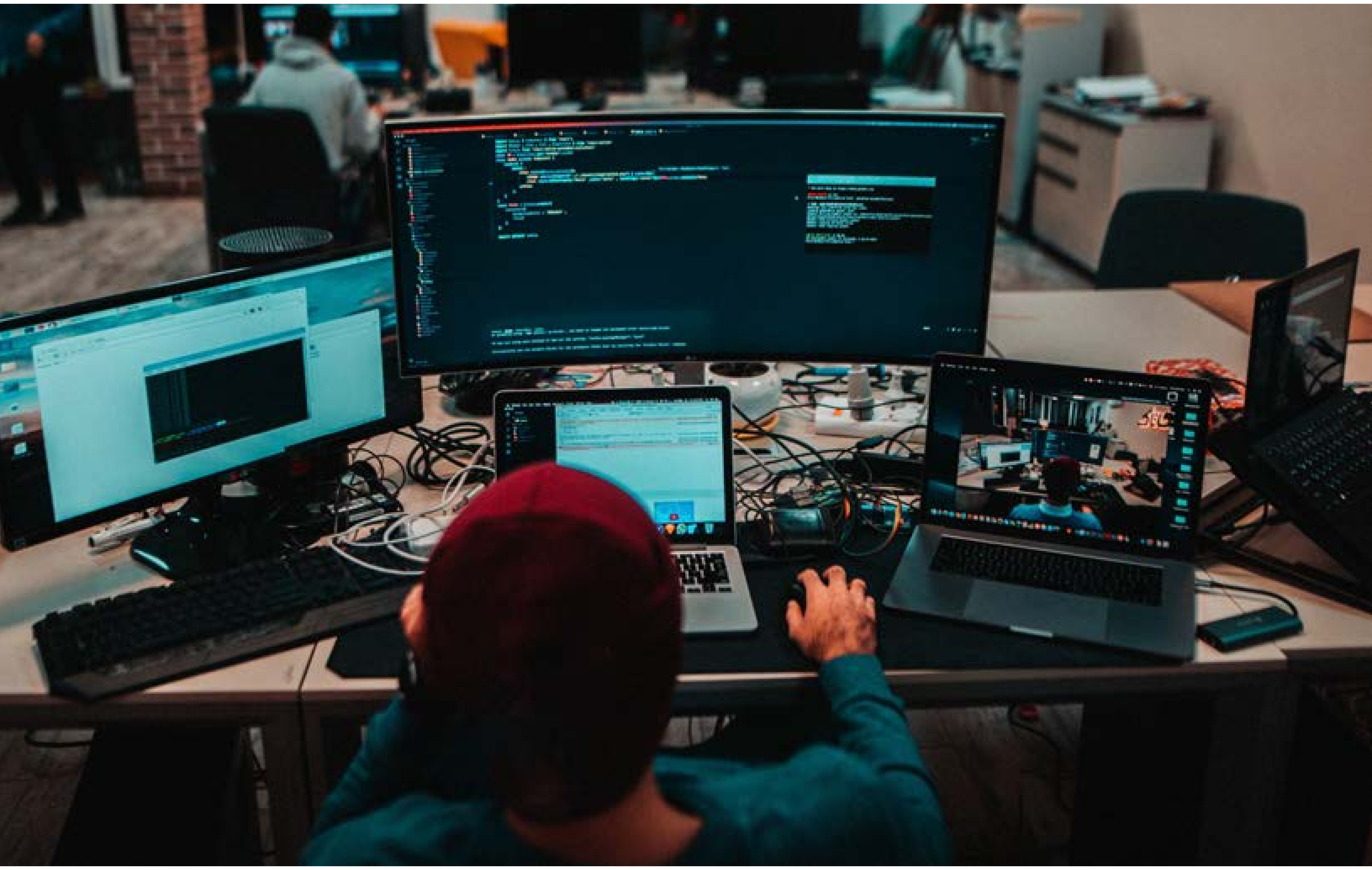
PROD PERFECT

WHITE PAPER



How to Maintain a Regression Test Suite

Dan Widing, Founder & CEO

TABLE OF CONTENTS

INTRODUCTION

"Being busy does not always mean real work… seeming to do is not doing."
—Thomas Edison

If you've recently gone through the gargantuan task of spinning up an automated browser testing suite from scratch, congratulations. It's not easy. It's a huge step toward deploying better code with fewer bugs, and toward achieving that coveted holy grail of software testing—Continuous Development (CD) with Continuous Testing (CT).

Now that you're done, the real work begins. You need to maintain this test suite over time. It needs to keep up with your developers; it needs to evolve with your application. If left alone, it will degrade, it will break, it will throw lots of false positives, and it will stop being used. All that hard work will fall apart.

This need to maintain and expand an automated test suite is why QA automation teams tend to expand alongside the growth of the dev team: There is an approximately fixed amount of work per hour of development. A common rule of thumb indicates a need for one QA automation engineer per approximately four developers.

There are three separate ways to think about stability, all of which are important and are approached differently:

1. Build-to-build
2. For stability
3. Over time

BUILD-TO-BUILD MAINTENANCE

With each new build, there is a risk of a user interface (UI) change breaking your test suite. If anything substantial about the UI changes, or the workflow being tested changes, the test will fail because it cannot find an element it's trying to interact with.

To keep these well-maintained, the fastest and simplest approach is to break the tests as early as possible. Sometimes teams attempt to have developers consistently communicate to QA engineers the UI changes coming down the pipe, but it's difficult to ensure consistency here. Furthermore, it's difficult to discern which hooks to use for a test until you see the application itself in a browser. Instead, if you simply test the application as early as possible in the development cycle, QA teams can investigate failed tests and fix them before the new code is deployed to production.

"if you simply test the application as early as possible in the development cycle, QA teams can investigate failed tests and fix them before the new code is deployed to production."

MAINTENANCE FOR STABILITY

Test instability (or flappiness, or flakiness) simply means that a test won't pass or fail consistently against the same application. This can be for many reasons: Frequently a test will time out due to network instability or a slow testing environment; test data can change; browsers are their own kettle of fish and tests can fail across different browsers even if there aren't bugs.

You don't want these tests breaking when it's time to find bugs. To test these for stability, it's best to run them hourly (or at a similar frequency) in a parallel environment to the deployment cycle, sending results only to the QA team. If they are passing and then fail without a new build having been deployed, you'll know there are stability issues, and can go fix them before they've raised alarms with the dev team.

MAINTENANCE OVER TIME

This is where things get hard. After building your regression test suite, your application will continue to change. In addition to brand new, significant features being added—where adding tests is not easy but the need for new tests is more obvious—features morph and small new features are added. Customer usage changes in ways not directly related to new features as customers discover new happy paths throughout your application. If you're not careful, your coverage will drop as the application and the testing suite drift from each other. There are in fact two risks: you'll leave important features uncovered, and you'll maintain tests that are of low value (this is called "test bloat").

Maintaining coverage of the most pivotal use cases or user flows requires data. You need to understand factually how your customers are using your application, and compare that information to your current testing coverage. We cover the data-driven testing process in this post, and specialize in driving test coverage using data as a business. We're passionate about data-driven end-to-end testing because getting it right solves a multitude of problems at once: your test suite is more accurate, more stable, and costs far less time and energy to maintain.

Test maintenance is an ongoing task that requires substantial attention; without that attention, all the hard work of building the browser testing suite will yield diminishing value until it's no longer paid attention to. The good news is that the hardest work is behind you, and with rigor and a good plan for each form of maintenance, the testing suite you just built will yield useful results for years to come.

"You need to understand factually how your customers are using your application, and compare that information to your current testing coverage."

ABOUT PRODPERFECT

Unleashing the power of machine learning to solve the hardest, most important, and previously unsolved problems in end-to-end (E2E) QA testing, ProdPerfect is the only autonomous E2E regression testing solution on the market that continuously identifies, creates, maintains, and evolves E2E test suites via data-driven, machine-led analysis of anonymous live user traffic. It is a fully-managed testing solution that addresses insufficient test coverage which causes critical and costly bugs in production; removes the burden that consumes massive engineering resources; and eliminates long test suite runtimes that slow deployments and decrease developer velocity.

SCHEDULE A PRODUCT INTRODUCTION TODAY

PRODPERFECT