PROD PERFECT

WHITE PAPER

From Bloated to Efficient

How to Actually Continuously Run E2E Testing

Dan Widing, Founder & CEO

TABLE OF CONTENTS

INTRODUCTION

Continuous testing is the holy grail of quality assurance when developing web applications. Continuous testing goes beyond preventing bugs from being shipped: it gives your developers immediate feedback on the small amount of code that would have just broken your app. And the immediacy of this feedback really, really matters. For developers, "context-switching" is anathema to productivity. Bouncing between different parts of the app is like trying to read multiple books books at once. But if the developer still remembers what they wrote, they can diagnose the bug much more quickly than if the bug was caught later and added as a ticket. Continuous testing means much faster and earlier bug resolution, and much higher developer productivity.

But supporting continuous testing at the browser level, where Selenium-like code is typically deployed for end-to-end (E2E) testing, poses particular challenges that require intelligent strategies to overcome.

"Continuous testing means much faster and earlier bug resolution, and much higher developer productivity."

CHALLENGES WITH CONTINUOUSLY RUNNING E2E TESTING

E2E testing is the most resource-intensive and unstable form of QA testing. Though teams have tried for years to move away from needing it, E2E testing remains the best way to know that your entire application will function for a user when they are trying to use it.

There are two primary challenges in maintaining an E2E test suite, and both are exacerbated when trying to run the tests continuously with every build:

Resource intensity.

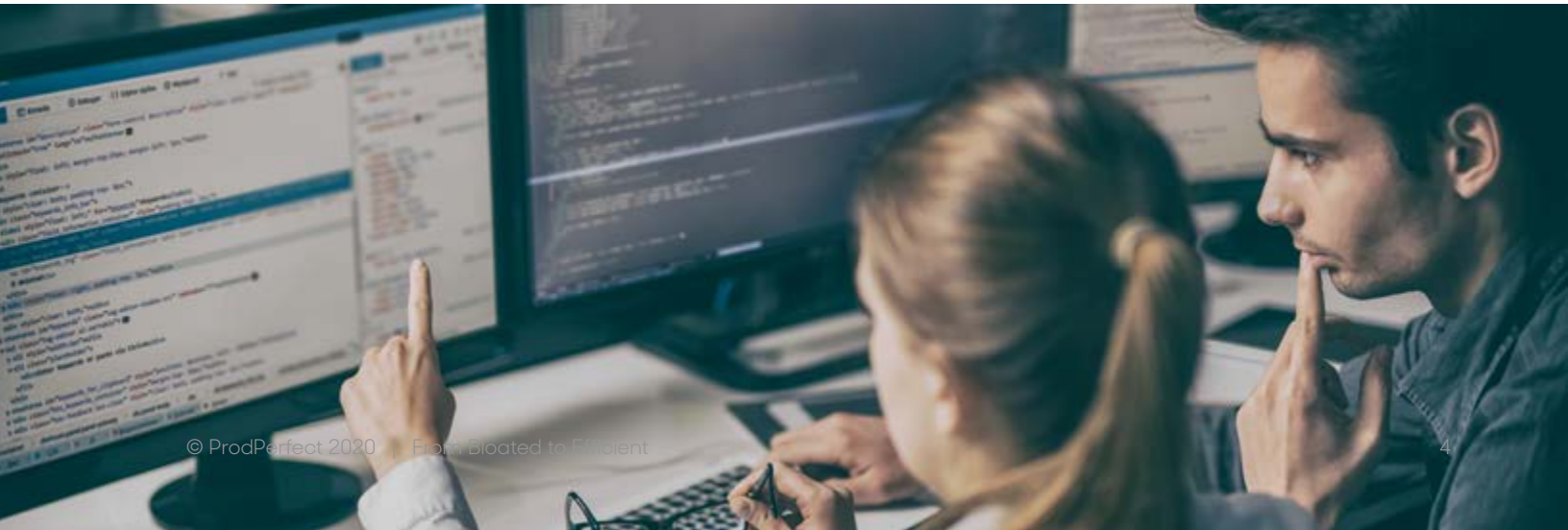
Because these tests are difficult and unstable, they need to have talented resources devoted to maintaining them. The testing team also needs to update the tests rapidly when they break (and they will break) to actually do the "continuous" part of continuous testing.

Runtime.

E2E tests take the longest to run (compared to lower-level tests). Long runtimes can also take the "continuous" out of continuous testing: it slows down the deploy process to the point that it cannot support frequent deploys.

When runtimes creep up (sometimes taking hours or more), E2E testing is often relegated to nightly or weekend runs, meaning any feedback comes not only long after the developer has shipped code, but long after users have seen it in production. Users, thus, have a buggy experience and developers now have to switch context to fix the bug. E2E testing clearly begins to lose its value.

There are hacky approaches to reducing runtime, including parallelizing tests or increasing testing environment performance to decrease wait time. But there's a much better way to reduce both runtime and total required resource allocation: reduce the number of E2E tests.



PICKING THE RIGHT TESTS TO RUN: FOCUS ON YOUR USERS

Fewer tests take less time to run and require less work to maintain—this fact alone is not revolutionary. But the reason most test suites become bloated is because of how test cases are retroactively developed: over time, test cases are simply added to the existing test suite, either in reaction to new features or in reaction to bugs. These tests continuously add to runtime and maintenance intensity until both become untenable.

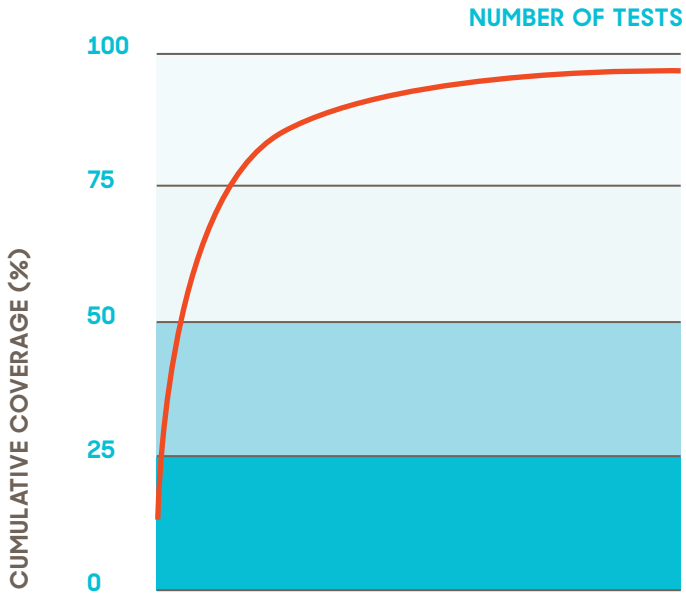
The ultimate reason this happens is that facts aren't driving decisions about which tests to write. Google suggests writing only a few E2E tests to cover what's most important. But how does one decide what's important?

> "It's simple: your E2E tests should be designed to cover what users are actually trying to do in your application."

If they can do what they want, they will be happy; if you need to reduce total test load to make continuous testing feasible, eliminate the tests that cover what users never or rarely do.

USING DATA TO DRIVE E2E TEST CASE MANAGEMENT

If we consider that each test covers some percentage of total user behavior (rather than simply a number of potential interactions), then we can imagine ranking each test from the highest to lowest incremental level of additional coverage of that actual user behavior. The result is an asymptotic curve:



The first few tests you add, on the left side of the curve, are highly valuable because they cover what the majority of users are actually trying to do. As you add more and more tests, though, their value becomes questionable, because they cover behaviors that few if any users are actually doing. Bloating the test suite with the tests on the right adds resource intensity and runtime, reducing both efficiency and total value. Building tests focusing exclusively on that left half, however, gets you the most bang for your buck, testing what most users are actually doing with lower engineering time and runtimes that support continuous deployment.

DATA-DRIVEN TESTING CORE STRATEGIES

Here are three essential strategies you can put into practice to focus your testing on what really matters:

Implement product analytics tools on production and use those insights to drive what to test.

This is where the facts come in. Let your users tell you what they're trying to do, and test those behaviors. Ignore what internal voices conjure up as a "best guess" in a conference room, and rely instead on real user data.

Frequently scrutinize your test cases to decide what to add and what to retire.

To prevent test bloat, go back to the well of product analytics data on a regular basis. Build a new set of test cases from scratch, and then compare your current test suite to those new test cases. Add what's necessary, and have the courage to remove what isn't. This will enable your test suite to keep up with the evolving application without becoming bloated.

Manage the test suite's runtime as a resource.

To be truly valuable, your E2E test suite needs to run continuously and quickly, so it's important to exercise discipline in your team about what's going to make it into the continuous test suite and what isn't. A timebox and disciplined limit is usually the best approach. If you really feel there are more tests that need to run than can fit into your predefined timeline, then split your suite into two: "core" and "edge." The core test suite will be run with every build and the edge suite will only be run nightly, catching bugs that affect far fewer users and therefore aren't as urgent when they do come up.

CONCLUSION

Following these strategies will allow you to keep runtime tight and maintain a highly effective test suite with limited resources. When you introduce an E2E test suite as part of a continuous development process, your dev team will love you for it, and they'll come to respect and investigate every broken test that comes their way. Your bug output will plummet and your developer productivity will excel.

Continuous testing with continuous development is testing nirvana: it's worth the extra strategic thought to achieve the highest level of enlightenment.

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

PROD PERFECT