PROD PERFECT

WHITE PAPER

Preventing Human Burnout

A Meaningful Approach to Measuring E2E Test Coverage

Wilson Funkhouser, CoFounder & Head of Prototyping

TABLE OF CONTENTS

TEST COVERAGE IN A LIVING ECOSYSTEM

I like to see any company's Quality Assurance (QA) as a living, breathing ecosystem. The ecosystem is defined by your business needs, the complexity of your application, and the innumerable ways in which you QA your system. Together, you, your developers, stakeholders, and customers all live in this ecosystem and vie for the free energy therein. In order to maintain ecosystem equilibrium, your company must balance each of these moving parts. Every ecosystem has rates of churn of their constituents. If you push your developers or customers too hard, they will burn out and leave the ecosystem. In this way, your QA ecosystem is as much about maintaining a stable application as it is about maintaining stable humans.

The QA ecosystem is what makes designing metrics for test coverage uniquely challenging. Each business has different needs for different types of tests, just as each business has a distinct web application and a unique user base. And because there are so many types of tests and ways to test, there are countless ways to measure adequate coverage. Despite this complexity, the basis for making measurement-related decisions remains largely unspoken:

Should we prioritize mere statistical coverage of code, features that are more important to our company's needs, or areas that the internal team deems more likely to break?

There is no single solution. If we want to measure end-to-end test coverage successfully, we must first identify metrics that are truly meaningful to our individual businesses and which reflect that individual humans are part of this system.

In distinguishing proper QA coverage metrics for your business, here are three key considerations to keep in mind:

1. Moving Fast
2. Having Acceptable Coverage
3. Covering Business Priorities

Moving fast.

Are you impeding your developers unduly? To move fast, we need to balance the cost of adding tests with the cost of repairing bugs and the cost of runtime in deployment. The more tests you write, the more stressful and time-consuming writing code becomes. The fewer tests you write, the more stressful and time-consuming maintaining code becomes.

Having acceptable coverage.

Since no QA system will realistically have 100% coverage for any application or cover every single use case, we must focus on the level of coverage we actually need, accounting for where the limited resources of our ecosystem are best allocated.

Covering business priorities.

As we measure coverage, we need to balance business-level metrics or key performance indicators (KPIs) with the ways that users are actually using the web app. If your sign up breaks for a few hours but your business model focuses on another conversion metric (such as adding items to a cart), this may not be the worst possible bug for you. It's up to each business to determine which test efforts directly yield ROI.

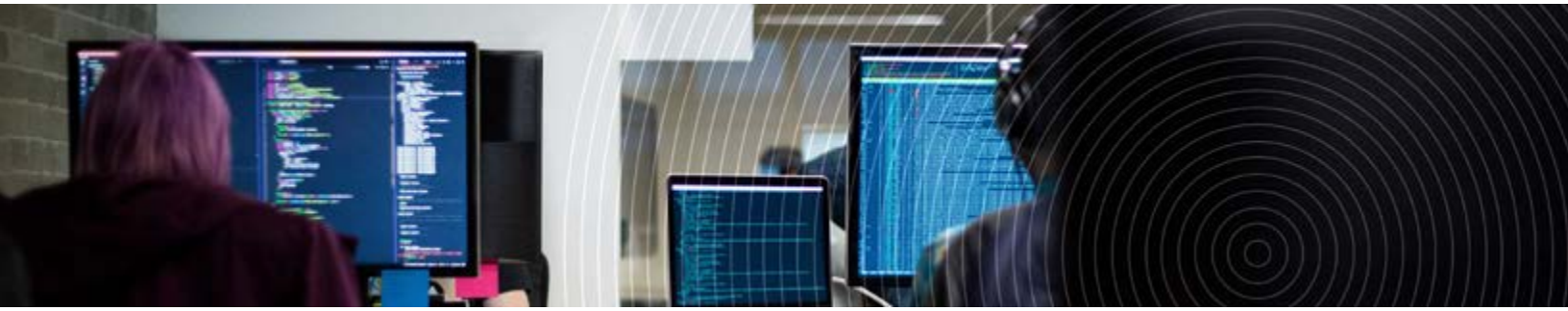
WHY DESIGNING QA METRICS IS PARTICULARLY HARD

Engineers have to manage complex product interactions in a changing ecosystem.

Worse yet, different types of testing often have different owners with different stakeholders who have different needs. This means managing variations in developer expectations, management needs, and in test suites. In the QA realm, since there is so much variation in what applications do, there are many ways to test:

- **Do you test at the unit and the controller level?**
- **Do you stub out or do you test calls to the database?**
- **Do you need to test your load balancing?**
- **And since the test ecosystem needs to remain dynamic, the test suite will change every time the product changes.**

Even if nothing changes other than an increase in the total number of users, engineers need to make sure that there are an adequate number of tests for load testing the site.



There are no tools nor individuals fully available and equipped to effectively measure coverage.

Product Owners tend to think of high-level features and development processes, then delegate responsibilities. Product Managers and QA Managers tend to focus on regression testing and manual testing. DevOps tend to own building test environments. And Product Engineers (and most engineers) tend to be the ones writing unit, integration, and API tests. There's no singular vantage point for fully understanding what has been covered. It gets even more complicated when thinking of feature complexity. Take the example of a login: there are innumerable ways in which the simple concept of logging into a website occurs. You can login from the homepage, from the checkout, or even through your email provider. Today, it's impossible

for a human to navigate this complexity. When I was a Product Owner, I remember reviewing the design of my web applications and realizing, "Oh my. No one is using this." Looking back, I'm ecstatic that I screwed up. It showed me that there are an incredible amount of biases and assumptions in how we think about applications we've built that simply don't align with the reality in how they're actually used on the ground.

Product Owners often don't have an effective means of managing incident response. Most of the time, when something breaks, the go-to response is, "let's write a new test for the bug we found and make sure it doesn't break again." This put-out-every-fire-as-it-comes approach to writing tests leads to a bloated test suite with hundreds of web regression tests and thousands of unit tests. This tactic may sound sensible, but in reality, regression testing must consider speed—not just of test suite runtime, but of human developer productivity. In the industry, we talk about burnout because of too many bugs, but what we don't talk about as often is burnout from too much testing.

We still lack an organized ethos for how companies can balance test coverage with other variables, including speed and fighting employee burnout.

To fully solve this problem requires fundamental shifts in how we assess our QA ecosystems.

TWO CONSIDERATIONS FOR MEANINGFUL DESIGN

Set a defined framework for meaningfulness.

Determine what should be tested in a way key stakeholders agree yields intrinsic value for your company. In addition to building an ecosystem which prioritizes preventing developer burnout, what's meaningful to us at ProdPerfect is focusing on a data-driven, user-based framework: we test what happens most—based on how users are actually using a site. We've defined a framework for covering a critical amount of everything that goes on on a given application. It ensures that the things people do most

"What's meaningful to us at ProdPerfect is focusing on a data-driven, user-based framework: we test what happens most—based on how users are actually using a site."

frequently on an app aren't going to break, while avoiding overburdening developers with maintaining bloated test suites. As an organization, we're devoted to minimizing the time needed to keep up with QA in order to prevent developer burnout and customer burnout from being exposed to too many bugs. Our principle is that just as we need to account for the burnout of having too many bugs, we likewise need to account for the burnout of having too many tests.

"Our principle is that just as we need to account for the burnout of having too many bugs, we likewise need to account for the burnout of having too many tests."

Determine an acceptable level of coverage when it comes to bugs.

Once the framework is defined, set expectations for what level of coverage is acceptable with respect to the impact of a bug when it reaches production. Is there a critical mass of users on certain parts of the webapp? Is some functionality hypercritical for customers getting value out of the product? For your internal KPIs? If you can answer the question of coverage acceptability in a way that allows you to have a sustainable business model and is backed by quantitative analysis, your employees will be more likely to maintain their zeal for their work, directly impacting the development of your product and the satisfaction of your customers.

A RUBRIC FOR COVERAGE

QA is a living, breathing ecosystem inhabited by humans. These humans are limited resources that can burnout from any number of factors. For this reason, the rubric should never be to have 100% coverage. The rubric always needs to consider the humans at the company and the humans using the app when deciding how much test coverage is meaningful.

When it comes to QA metrics, there are no right or wrong answers. There is simply the question: "Is your ecosystem stable and sustainable?"

For some companies, stability means: Yes, we will burn out developers. They will spend a year here and leave, because we are writing so many friggin' tests. For ProdPerfect, because we prioritize maintaining a balance between the three considerations of 1) moving fast, 2) having acceptable coverage, and 3) covering business priorities, we're empowering both ourselves and customers to let the right things break and stop the right things from breaking. And we're going to keep building on whatever parts we can in order to meet changing needs in the changing ecosystem of QA, humans included.

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