PROD PERFECT

WHITE PAPER

Beyond Whack-A-Bug

Shifting From a Reactive to Proactive E2E Testing Strategy

Erik Fogg, CoFounder & Chief Revenue Officer

TABLE OF CONTENTS

INTRODUCTION

Of all of the classic arcade games, Whack-A-Mole just might be the most frustrating. You can't win the game of Whack-A-Mole. Every time you think you've hit the mole, the little scoundrel always finds a way to pop up again somewhere else, and you're always one step behind.

In the world of end-to-end (E2E) testing, we can get stuck playing Whack-A-Bug when we reactively write tests to bugs that pop up in production in order to prevent them from appearing again. I like to call this practice Whack-A-Bug testing because it's a common approach, yet can easily become an endless cycle in which testers are always one step behind. As leaders in our organizations, we must instead figure out how to be one step ahead, committing to developing more courageous, forward-thinking E2E testing strategies.

WHY WHACK-A-BUG TESTING IS SO COMMON

Quality Assurance (QA) teams live in an unenviable world: their work is largely unnoticed until something goes wrong. They're then faced with explaining their decision not to test something, especially if the results cost the organization money. We've heard so many times, "Why didn't this get tested?" Whatever the reason for the revenue-impacting bug, if QA "fails" to assure quality, they can be chastised by other leaders in the company. QA leaders and engineers are thus under immense pressure to prevent mistakes and ensure the same bugs don't pop up in production more than once.

"Unless you're committed to testing every possible permutation of behavior, it's essential to re-prioritize."

Prior to co-founding ProdPerfect, I used to do some consulting work with factories. Operations such as oil refineries are able to clearly track metrics related directly to revenue and cost, and can thus fairly easily make decisions related to return on investment (ROI). In a refinery, you can easily measure how many barrels of oil were produced each day, and thus prioritize what investments to make to increase this daily number. Many factories lose sight of the ROI on their actions and become reactive to the most recent problem that occurred in the plant. But when they do make the commitment to prioritize their work on high-ROI improvements, it's fairly easy to measure the results.

When you measure performance of software QA within an organization, you can't measure six man-hours of code in barrels. Test code isn't uniformly valuable, and each update doesn't have an equal impact (good or bad) on the business. Because of this variance, it can be hard to resolve multiple competing priorities: **How do we deliver value and quality without sacrificing speed?** The lack of clarity in measuring speed, value, and quality prevents engineers from having clear ROI agreement and unified productivity KPIs.

Despite the challenging and often organizationally unclear work of measuring performance in engineering, the pressure to perform still exists for engineers. And since QA teams often feel the brunt of this pressure, QA

leaders are often forced into operating from a reactive stance in their E2E testing strategy. In other words, due to a lack of clear prioritization metrics, many teams resort to playing Whack-A-Bug with their testing to appease the organization at large. Sticking to high-ROI initiatives in testing is difficult, but just as important as in a refinery.

"The fallacy of Whack-A-Bug testing is assuming that if we create another test where we previously saw a bug, then we're more secure than if we wrote that test elsewhere."

THE FALLACY OF WHACK-A-BUG TESTING

To challenge this practice, we as leaders first need to ask ourselves: Does a bug slipping through in one part of an application increase the likelihood that it's going to happen again in the same place more than anywhere else? I like to think of the comparable, yet common fallacy in gambling: If you pull the slot machine six times and didn't get jackpot, does this mean the next time you pull it, there's a higher likelihood of hitting jackpot? Though our guts may tell us otherwise, the answer to both questions is ultimately no.

The fallacy of Whack-A-Bug testing is assuming that if we create another test where we previously saw a bug, then we're more secure than if we wrote that test elsewhere. It's simply not the case that because a bug happened

in a certain area, then it follows that there's an increased likelihood that the bug will happen in that area again. Whack-A-Bug testing is NOT a proactive testing strategy: a Whack-A-Bug test doesn't add a test to the area of the application with the greatest need for tests. Instead, it is a passive strategy: we're testing an area as a reaction to seeing a bug there. The fact that a bug came up last week shouldn't change our organizational focus on writing tests for high-priority areas: areas that are more likely to produce bugs, that are important for customer use, or that directly impact revenue.

WHY WHACK-A-BUG TESTING HURTS MORE THAN IT HELPS

Whack-A-Bug testing hurts engineering organizations for several reasons:

It distracts us from writing well-prioritized tests.

Before any bug shows up in production, a QA team has developed a strategy for what to test, based on a certain mechanism of prioritization. When we write Whack-A-Bug tests, we're pivoting our test-writing resources away from whatever prioritization mechanism we otherwise had and towards the Whack-A-Bug test. As a result, we delay writing future high-priority tests.

It adds maintenance burden to your team.

Whack-A-Bug testing decreases your team's capacity to write future high-priority tests. Every time you write an E2E test, you're committing to maintain that test. This creates a fixed, unavoidable ongoing level of work that decreases your capacity to write more tests with the same number of engineers. Most teams I've seen attempt to make up for this by simply hiring more.

It slows down developer productivity and velocity.

In a continuous delivery process, each new E2E test adds to the test suite run-time, which lengthens your regression cycle. If your test suite takes half an hour to run and you're running with each commit, this means either that 1) your developers aren't producing anything during that time or 2) your developers are checking in code less often because they don't want to wait for the tests to run (or both). In both cases, you lose developer productivity and provide less-frequent quality feedback for each build, meaning each incremental Whack-a-Bug test costs developer velocity.

It bloats your test suite and increases instability.

At some point, your test suite will have grown large enough that there's a high probability that it will fail on a given run due to instability. Once instability reaches a certain critical mass, the test suite fails so frequently that developers stop paying attention. When that happens, it starts providing negative value: adding deployment runtime without contributing to quality.

REPAIRING THE DAMAGE OF WHACK-A-BUG TESTING

How do we reverse the damage of Whack-A-Bug testing? First, our organizations need to re-examine our testing choices through a blank-slate exercise. We must ask ourselves: **If we were to build this strategy from the ground up once again, what would be our testing priorities? What would we test to balance test coverage with speed, maintenance burden, and stability?** Once we've defined what's ideal for us to test, we need to then overlay that outlook on what's currently being tested as is.

And here's the hard part: we need to have the courage to shut down tests that don't align with this strategy. And we need to move on.

One helpful aid in this process is to evaluate tests that have been in the suite for 6 months or longer and review: Have they caught any bugs in the last 6 months? If they haven't, your team should strongly consider retiring them, as they're likely not worth prioritizing. Unless you're committed to testing every possible permutation of behavior, it's essential to re-prioritize. What we learn by doing this exercise is that most Whack-A-Bug tests never actually catch a bug. Whack-A-Bug tests may give us short-term comfort in the face of organizational political pressure. But when we let the data speak instead of human impulse, we see that the vast majority of the time, writing Whack-A-Bug tests provides little to no real business value. It's when we stare this harsh reality in the face that we find the courage that we need to reprioritize our test suites, drastically reducing the number of unnecessary tests.

"It's when we stare this harsh reality in the face that we find the courage that we need to reprioritize our test suites, drastically reducing the number of unnecessary tests."

Ultimately, developing a new testing strategy from the ground-up that highlights your most important priorities will free up your developers and QA resources to properly cover what's truly important to your business.

The benefit is three-fold:

1. Better Quality
2. Better Speed and Cost
3. Higher Trust From Your Developers in the Test Suite

SHARING A COMMITMENT TO BETTER QA

Many QA teams resort to Whack-A-Bug testing because they're under pressure to respond to quality problems in prod. A better QA practice is only possible when all leaders across our organizations share and fulfill the commitment to stick to the team's QA strategy, rather than muck with it every time something seems to go wrong.

First, it's crucial for engineering and product leaders to recognize alongside QA leaders that Whack-A-Bug testing does not necessarily improve quality assurance. We must understand that a bug appearing in a certain place shouldn't necessarily change priorities for what to test moving forward. Our leaders must keep their commitment to a well-defined trade-off mechanism between speed, productivity, and QA. Each organization's trade-off will be different and change over time, but it needs to be sacred at any given time.

When we see a bug in our code, we need to commit to asking:

- **Do we need to rethink our strategy, or possibly rethink our trade-off point?**
- **Are we prioritizing the right way?**
- **Are we making the right commitment to what an acceptable level of testing looks like?**
- **How might our testing approach be unduly burdening our team?**

This discipline helps us resist the knee-jerk reaction to build a new Whack-a-Bug test. At ProdPerfect, our commitment to each other is that we will not be reactive in determining QA priorities.

"At ProdPerfect, our commitment to each other is that we will not be reactive in determining QA priorities."

We invite you to join us in this commitment: We will not act upon knee-jerk reactions to immediately build tests for every bug. Rather, we will learn from bugs. We will evaluate them over a period of time by overseeing where bugs are slipping out and what damage they're producing. Then, we'll make data-driven decisions to inform our testing strategy. We'll collectively own the consequences and costs of making changes to our testing strategy. But we will not knee-jerk respond by playing Whack-A-Bug with our testing approach.

Making this commitment requires organizational discipline and courageous leadership from all. All our leaders must agree to see quality assurance as a partnership in which organizations need to effectively balance their testing priorities and determine the best level of coverage for the business. Each of us has something to benefit in making such a commitment, and I invite you to share in this commitment with me. Instead of burdening our processes and teams with reactive Whack-A-Bug testing, let's care for them well by thinking proactively and letting data alone drive our testing strategy.

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