New Metrics of App Testing

Pass/Fail Tests Aren't Enough;
Metrics Must Be Analyzed Throughout Production for Winning Apps

WHITEPAPER

Introduction

As mobile and desktop advance and more people weave technology seamlessly into their everyday lives, how applications function and how they're created, tested and maintained needs to change. We've entered a time where people expect more from their technology. They expect applications to function correctly, run smoothly and not bog down their devices from day one. If there is a bug, your users will go elsewhere – they have plenty of options and little patience. Companies now need to perform deep metric analysis and tweak their apps constantly to ensure apps delight their users in every way and provide the best experience possible. Enter continuous testing, Testing in Production (TiP) and the rise of modern application metrics.

Adopting a mind-frame that embraces continuous testing and Testing in Production will help teams produce higher quality apps in-line with constantly shifting user demands. It will also provide them with the metrics that are most important to modern development and real-world success.

Testing in Production

Testing in Production (TiP) is a testing method that advocates releasing your product to the public while having developers on hand to monitor and immediately fix any bugs. Bear in mind that this method will not work for applications that have a vetting process (such as iOS mobile apps). It may

seem like a risky option, but if done correctly, TiP can be extremely useful. Testing in Production does not mean releasing an untested app into the hands of your users. Today's users have high expectations for app quality and releasing an app with major bugs will sink you before your team has a chance to address the in-the-wild issues. Instead, your app should have already gone through traditional in-house testing and in-the-wild testing to find the most important bugs. TiP can then be used to find additional fringe use cases or bugs that appear rarely and only on certain devices.

Continuous Testing & TiP

We've reached a point of technology saturation that means companies are continuously releasing new versions of their applications. Whether you are using a continuous release method or are working to complete a new version in its entirety before release, employing continuous testing helps ensure your application is as bug free as possible when it hits end users. Testing throughout development gives teams more time to find and fix bugs and helps prevent bugs from effecting later code.

Testing in Production (TiP) is a natural extension of continuous testing. By the time you near launch, your product should be largely bug free from your previous work. This is what makes TiP feasible. Though Testing in Production involves releasing your application to the public, keep in mind that this can be accomplished through a limited release – either to a subset of users or during a lull in activity. This gives you the same end user insight while limiting exposure of a potentially faulty product. Since the product is in active use, development teams have the ability not only to find bugs that eluded them in the lab, but also see if a bug fix works almost immediately. This last line

of testing defense helps developers find and fix issues quickly, before they have a major impact.

Another advantage of TiP is that it's a great way to gather data from real life use conditions. When testing in production, you see how an application works for a real user, on their actual device, under real-world conditions. The only way to replicate this kind of insight without releasing your application to the public is to test in-the-wild with a crowdsourced testing company. This window into real life use not only reveals bugs missed in the lab, it also provides teams with the modern metrics they need to create the best app possible. By watching how an application functions for real end users, teams are able to gather and act on real metrics to optimize applications before they reach a larger crowd.

> *"It's no secret that products like Bing, Amazon, Google and Facebook launch experiments all the time exposing code and features to a small number of uses. This is a powerful way to get information on your product in real-world conditions that simply cannot be reproduced in a test lab."*
>
> -Seth Eliot
> Senior Test Engineer, Test Excellence
> Microsoft

TiP Methods

Testing in Production requires a careful balance between not disturbing your uses too much while getting the new release in front of enough eyes to

generate useful information. Here are a few methods (identified by Seth Eliot, a Senior Test Engineer of Testing Excellence at Microsoft) that will give you an idea of what you should look for and what you can achieve practicing TiP.

- *Data Mining: Data mining allows you to track real usage patterns and tease out defects. It also lends itself to optimization since you can look at statistics from before and after changes. Another option is to collect the data in real-time for team members to analyze later. This can influence future changes.*
- *User Performance Testing: This issue will come up again when we discuss which metrics are most important. As far as testing goes, use TiP to get an idea of how your app performs across the hardware/software matrix. Like with data mining, this gives you access to real life results from a user's perspective.*
- *Environment Validation: You can run environment validation during the initial launch or collect data continuously. This category involves traditional pass/fail tests that look for version compatibility, connection health, installation success and other important factors across the user matrix that are hard to replicate on a large scale in a lab.*
- *Experimentation for Design: This is your typical A/B test. Divide your users into two (or more) groups and give each a different design to see which users respond to best.*
- *Load Testing in Production: Release your application into TiP and add synthetic load on top of the real users. You'll be able to see exactly what happens if something goes wrong when your app is hit with heavy traffic – before you disappoint a real tidal wave of visitors. Load testing this way can help you identify issues that may not appear in traditional automated load testing – such as images not loading properly. Be careful, however, to not affect your app's load time too greatly while testing or you risk driving away the real users.*

Ideally, these tests shouldn't have major adverse effects on users – most major issues should have been caught already. Many of these tests can be performed traditionally, in ways that don't involve your real users. With that being said, it's important to remember that your real-life audience will be the ones who actually use your application. So do both. Use traditional testing methods initially, then use TiP as a sanity check. The first phase (the continuous testing) helps you spot important bugs while the second phase

gives you the most accurate and useful information – actionable metrics from real users that you can't get from a controlled environment.

The Metrics

In addition to traditional pass/fail testing, performance based metrics, especially when gleaned from real-life devices, are particularly important these days. Users will quickly abandon your application if it is slow to load, uses too much memory or doesn't interact properly with other aspects of their device. An end user doesn't care about how many test cases passed or failed, those are useless metrics in the long run. Instead, you need to consider metrics like CPU usage, API performance and system response time while testing. These are the things your users really care about.

As technology continues to become more pervasive, everyday users will get savvier and more comfortable with the "tech" part of technology. There are already a number of consumer-facing applications that help users measure their connection speeds, data usage and a variety of other information that used to be strictly in the realm of developers and testers. With the glut of big data flooding in, it is helpful to focus specifically on the metrics users themselves can access. These will be the ones they are paying attention to and the ones that will be influencing their use habits. In many cases, ignoring these metrics can cost you users – and ultimately revenue.

End Users

What's the point releasing an application if no one uses it? Testing should be extremely end user focused. After all, they're the ones who will ultimately be making you money by using your application. Understanding your end

user can help contain testing costs and looking at metrics from an end users' point of view will help your app succeed. Always test with your end users – and their sentiments – in mind. Remember, waiting six seconds for a page to load doesn't seem long on paper, but according to the majority of users, it's long enough for them to give up on your app. This is the mind frame you want while looking at metrics.

Don't waste time testing on devices your target audience likely won't be using. Likewise, don't only test one device, operating system, browser, etc. Your users will be on a multitude of hardware/software combinations and bugs are guaranteed to slip though if you don't test on as many of them as possible. Identify the most common combinations within your target demographic and begin your testing with those. This is another instance when crowdsourced testing or Testing in Production comes in handy. Testing with your actual users is a way to be sure you've covered the most important matrix considerations.

CPU

Using too much processing power is one of the biggest reasons people abandon applications. You absolutely must measure this. Slow response time is an indicator to users that something is bogging down their system and there are apps available that let users see how much power each application is using. If your application is flagged as the biggest abuser, users will likely look for a better-tuned replacement, particularly if your app isn't absolutely necessary. Your app isn't running in a perfect environment. It's competing for space and processing power with a collection of other apps. If your app requires too much effort to run, it'll be the first to go, or it may not even work at all.

Another important factor to remember – and another reason to monitor CPU usage – is that not all devices have the same processing capabilities. By not monitoring CPU usage across a range of devices, you may miss some major device-specific issues. It is important to particularly monitor and test CPU usage as you release new versions, or new popular handsets hit the market. Do not assume that because your application was working fine at one point that it will always be fine.

API

API requests can also have an effect on the response time of an application. If not properly integrated, APIs can slow down an application or completely fail to return a desired action. Whether you create a custom API or use an open source version, be sure to test the functionality and security of the API itself in addition to testing how it works on devices.

Though APIs are now common, the technology is less understood by the common user. This should be an even bigger incentive to carefully vet and monitor any API requests integrated into your application. Users will recognize that something is wrong, but they won't be able to pinpoint the problem, which will lead to general frustration and anger.

Once you move out of initial testing and into a production environment, continue to monitor your APIs to ensure outside factors don't adversely affect the application's quality or the effectiveness of the API's service.

System Response

Testing CPU usage and APIs requests isn't enough to ensure good system response time – there is still a lot more to look at specifically related to system performance.

Measuring system response time has its own set of sub-metrics. Nexcess, a web hosting company, highlights these specific measurements:

- *Payload: The total size in bytes sent to the test app, including all resource files.*
- *Bandwidth: The minimal bandwidth in Bps across all network links from client to server.*
- *AppTurns: The number of components (images, scripts, CSS, etc.) needed for the page to render.*
- *Round-Trip Time: The amount of time (milliseconds) it takes to communicate from client to server.*
- *Concurrency: The number of simultaneous requests an app will make for resources.*
- *Server Compute Time: The time it takes for the server to parse the request, run application code, fetch data and compose a response.*
- *Client Compute Time: The time it takes for the application to render client-facing features.*

When testing system response, be sure to look at how long an action takes to complete from start to finish – from the second a user hits a button to the second the app finishes loading. Is the app usable before it's completely loaded? If so, which features are available first? Are they useful ones that will keep your users happy or secondary features that will only remind uses that they're still waiting for the real deal?

It is imperative you monitor system response under real world situations. Tests that only look at server-side response won't account for real use factors that can drastically increase the time it takes for the action to complete in the user's eyes. Some information may not return fully, information cached in browsers could have trouble loading, some data centers may have slower response times than others, a variety of things could go wrong that you won't know about unless you are monitoring real users in real situations.

Tips

Now you know what to monitor once your application is released to the public, but what if you turn up a problem? Here's a few recommendations that will help you process the data you're collecting and troubleshoot issues if they arise.

> *"Billions of dollars in revenue are lost each year by websites that are unable to measure and control client perceived response time. Clients get frustrated and leave websites before completing a transaction, often never to return."*
>
> -Software Systems Laboratory, Columbia University

API Issues

The best way to avoid API issues is to reduce API request complexity. Couple as many queries as possible instead of sending many individual requests. Similarly, if you are working with an application that has caching capabilities, figure out which items you can cache to cut down on the amount of data that needs retrieving every time your app is used.

It's also extremely important to remember that APIs can be affected by platform version, particularly within the Android ecosystem. Each version of the Android operating system only supports specific API classes. Identify the most popular devices within your target demographic and see what platform versions those devices support – it is often not the most recent platform release. Tailor your API integration to work with the dominate platform version (and remember to update as the new versions disseminate).

System Response Best Practices

How you measure system response time can drastically affect the data. Here are a few tips for collecting system response data that will most closely resemble what your users are experiencing.

- *Work with percentiles, not averages. Taking a broad measurement and finding the average response speed will not give you an accurate portrayal. This practice disregards the top and bottom speeds (important information) and doesn't give you any idea of how many users are experiencing what speeds. Measuring response speed and dividing the data into percentile designations will give you a clearer picture of the dominate response speed. If the biggest percentile has slow response times, you have an issue.*
- *Not all performance data is the same. Don't lump initial load time with response time for logging in – they are two separate actions and should be analyzed as such. One action may be slower than others (especially if an action relies on API calls) but if you look at all response time data together you will not know which action needs addressing.*
- *Cache what you can. Like with API requests, caching whatever data you can will reduce response time.*

Conclusion

Understanding that these metrics are important – and often accessible – to your users is a vital part of modern application development and testing. You cannot push aside data like CPU usage, API response and system response time and deal with it another day. Users will notice the issues and know that it isn't a price they have to pay for the technology. They can – and will – go elsewhere.

But don't be overwhelmed by the flood of big data. Take advantage of testing methods such as continuous testing and Testing in Production to help you not only find and fix bugs, but see your application from an end user's

perspective. Meaningful metrics are more important than a mountain of data. Knowing what deserves attention and what's just noise will help you focus going forward. These practices will become even more important as people continue to become more involved with their everyday technology and as more metrics become available.

About Applause

Applause empowers companies of all sizes to deliver great digital experiences (DX) – across web, mobile and IoT as well as brick-and-mortar – spanning every customer touchpoint.

Applause delivers unmatched in-the-wild testing, user feedback and research solutions by utilizing its DX platform to manage communities around the world. This provides brands with the real-world insights they need to achieve omni-channel success across demographics, locations, devices and operating systems that match their user base.

Thousands of companies – including Google, FOX, Best Buy, BMW, PayPal and Runkeeper – rely on Applause to ensure great digital experiences for their customers. Learn more at www.applause.com.

Americas Inquiries

Applause U.S. HQ
100 Pennsylvania Ave
Suite 500
Framingham, MA 01701
1-844-300-2777

Europe Inquiries

Applause Europe HQ
Kopenicker Str. 154
10997 Berlin, Germany
+49 30 57700400

Israel Inquiries

11 Galgaley Haplada
1st Entrance - 2nd Floor
Herzliya, Israel
+972.74.701.4240