rainforest

Scaling Mobile QA without Scaling Your Team

Why Scaling Mobile Testing Matters

The mobile application industry will generate over $77 billion in 2017, with almost 2 million apps in the Apple App store and 2.2 million apps available in Google Play.[1] However, a study by Nielsen found that the average smartphone user only uses about 26.7 apps per month, forcing developers to provide stellar user experiences to stay at the front of this very crowded market.[2] Offering a high-quality user experience and delivering the latest features quickly is critical.

Because traditional QA practices predate the existence of mobile apps, they aren't tailored to the unique challenges of testing mobile apps. These methods are often slow, clunky and poorly suited for the fast-moving mobile app teams. As a result, many teams with hybrid or mobile products haven't found a QA process that fits their product and workflow requirements, much less one that accelerates development.

In this guide, we'll walk through the techniques that teams can use to scale up their mobile app testing strategy without adding unnecessary headcount to their team (or more phones to the drawer).

[1] The 2017 Mobile App Market: Statistics, Trends, and Analysis, Business2Community.com
[2] So Many Apps, So Much More Time for Entertainment, Nielsen.com

In This Guide

Section One

What Makes Mobile Testing So Challenging?

Most modern QA testing processes are challenging, but mobile application teams face unique problems that contribute to a more complex, slower testing process. Understanding the challenges that developers face when testing their applications provides context needed to create testing processes that circumvent these issues and allow them to ship features faster.

Application Complexity

Mobile applications often feature complex user flows. They may be marketplaces with multiple key users in every flow like Uber, which requires testers to switch between multiple personas (or devices) throughout the course of the test scenario. They may rely heavily on third-party integrations or hardware components to function properly. A testing strategy for an application like Instagram must factor in compatibility with social media apps as well as the camera on the device being used. These complex use cases inflate the amount of work required to adequately test the application.

Deployment Constraints

App markets like the Apple App Store and Google Play represent an additional barrier to deploying apps into production. Every application must be reviewed to ensure that it meets the quality standards of these vendors, so a single overlooked bug might delay an app from reaching customer hands. Updates and bug fixes must also be reviewed by the distributor, which can significantly increase the time required for updates and patches to reach users. A development team must factor in these constraints when they consider their timeline for getting their app to market.

Device and OS Fragmentation

Mobile applications must provide a seamless experience across a huge range of devices and OS versions. New devices and OS updates are introduced all the time, requiring developers to be highly responsive to any changes to the hardware and software that may create issues for their users. Teams must account for everything from network carrier settings to battery charge levels. Fragmentation, the phenomenon that occurs when some mobile users continue to use older versions of an OS or device even after new ones are introduced, complicates manual testing.

Android Device Fragmentation

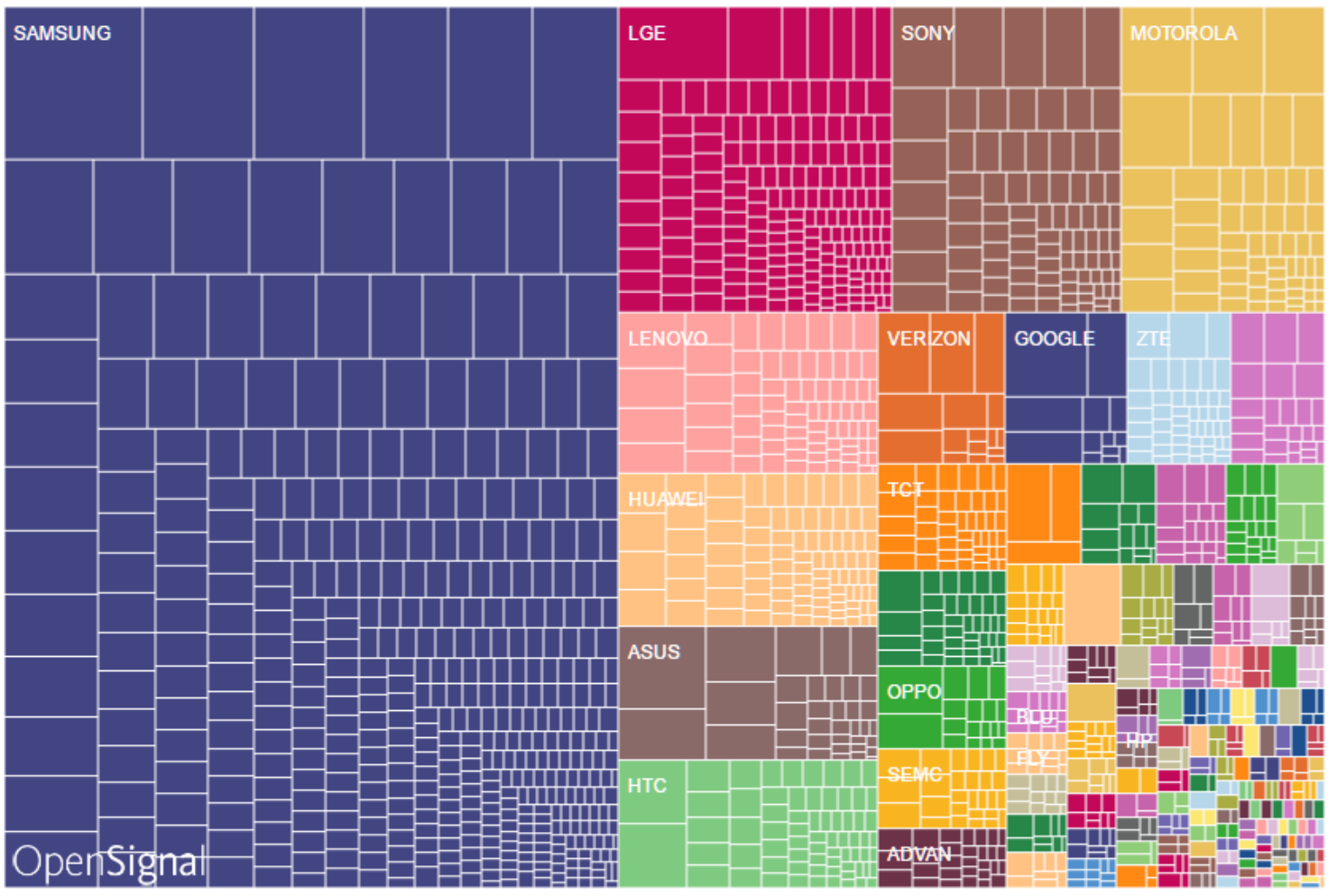


Image Source: OpenSignal

Testing every combination of device, OS and network settings creates a huge number of test cases. This requires development teams to perform the work of sourcing and maintaining of a growing pool of mobile devices. These challenges represent major roadblocks for mobile app developers. By using Continuous Testing techniques to build a flexible, fast-moving and reliable QA process the impact of these challenges can be minimized, giving developers more freedom to move faster.

New to Continuous Testing? ***Learn more here.***

Section Two

Scalable Mobile Testing Tools and Environments

Speeding up mobile testing requires some deliberate tooling and processes that bring bigger, better testing capacity where you need it most. Every team's setup will vary, but the need to develop the right testing workflow and maintain the right test environments are universal.

Testing Environments: Are Real Devices Necessary?

There's no getting around the fact that developers must ensure that mobile applications work well on user devices. The conventional method for testing mobile apps is to run through all test cases on devices in hand. But this is unsurprisingly a very hard practice to scale. Testing on real devices often means sourcing and maintaining a pool of mobile devices that reflect your users' preferences. Running through a full suite of mobile tests manually is a major time sink.

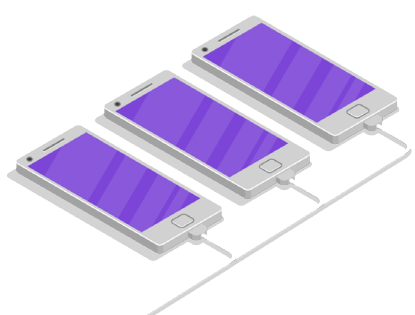
Device labs like Amazon Device Farm provides a scalable alternative on-demand testing without requiring in-house device management. Additionally, rethinking when you really need on-device testing and when a test can be run on a virtual machine can speed up testing significantly while keeping costs down, especially in the earlier stages of development.

Using Real Devices Efficiently

Even if you rely heavily on automation, crowdsourcing or device farms to execute mobile tests, it's beneficial to maintain a few key devices in-house, focusing on device and OS combinations that are most popular with your user base. The key to keeping real device testing from slowing down your process or eating up budget is using them only where the alternatives don't work. ***Reserve device-in-hand testing for critical UX testing and to cover flows that virtual devices and device farms don't replicate well***, including interrupts (SMS, other apps, calls, alarms) space limitations, sound recording and playback.

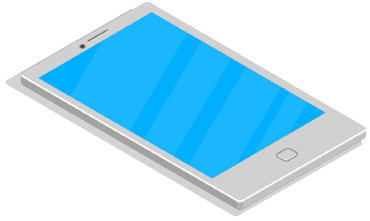
The Right Tool for the Right Job

QA teams often suffer from using resources inefficiently. Take inventory of the people, tools and processes that you're currently using to execute mobile tests, and think of how they can be optimized for better results. Here is a survey of the mobile testing tools available, and how teams can best utilize them.



Device Farm Testing

Device farms offer access to a large pool of real mobile devices remotely, offering the ability to test key functionality quickly and at scale. Use device farms to execute key functional and regression tests that are difficult to automate, and where testing on an actual device matters.



Device In Hand

Testing on real devices in hand provides the highest fidelity user experience testing. Because manual device testing is relatively slow and requires the greatest resource investments, using this sparingly to run hardware tests and ensure final quality before release.

Virtual Machines

Emulators can perform functional tests rapidly during development, when you need fast feedback most. Make virtual machines your testing workhorse to execute the bulk of your repetitive functional tests, especially during development when the application changes rapidly.

Testing Automation

Automated testing is the gold standard of execution speed and long-term costs -- as long as automated test cases aren't brittle or flaky. Use automation to streamline stable tests, especially in production.

Section Three

Rethink How Mobile Testing Happens

Testing is frequently delayed until an application or feature is nearing completion. But siloing QA processes at the end of development just creates lag between the creation and discovery of issues, reducing efficiency. Rethinking where QA lives in your organization and how it fits into the development workflow are the first steps to scaling up mobile testing.

A Workflow-Integrated Testing Process

The fastest mobile teams are those who have successfully pulled QA testing out of the final stages before deployment and integrated it into the entire development process. Scaling mobile testing without adding time or people requires integrating more effective testing directly into your existing workflow. Scalable mobile testing starts with continuous integration. Unit tests that are run frequently and consistently provide a baseline for code quality as early as possible, ensuring that minor bugs don't snowball into bigger ones.

Beyond using a mobile-friendly CI, ensure that your entire testing workflow is as "pluggable" as possible. For example, being able to get test feedback and bug alerts directly in your team's communication channels (like Slack) helps surface issues quickly and reduced the need to switch between different tools and platforms to check for issues.

Designate Mobile Testing Ownership

Many teams struggle to incorporate QA in development because testing lacks ownership. Mobile testing in particular can be a resource-intensive endeavor. Without a dedicated owner, it can fall short of the mark. As more teams adopt a developer-owned testing strategy and use more hands-off test execution methods like crowdsourced testing and automation, they find it necessary to have have a strategic member of the team who focuses on facilitating these processes.

Whether QA operations is a function of your DevOps team, a product manager or a dedicated QA engineer, integrating it into your development process strategically helps accelerate and optimize the testing cycle, especially at scale.

Make Quality a Team Sport

Formally testing every possible scenario is impossible, which is why empowering every member of your team to report bugs and participate in the quality process is critical. Dogfooding doesn't replace a strategic quality assurance process, but it can help surface bugs faster when utilized in parallel with more structured testing techniques. Dogfooding increases test coverage by emulating a wider range of real-world use cases. This is especially useful for testing mobile applications, which must work equally well across a variety of networks, devices, OS configurations, and usage scenarios.

An important step in a strategically managed dogfooding practice is to give the team a way to communicate any issues they find; having a process in place to document any issues that are surfaced through internal use ensures that bugs don't slip through the cracks.

QA Snapshot: Creating a Culture of Quality

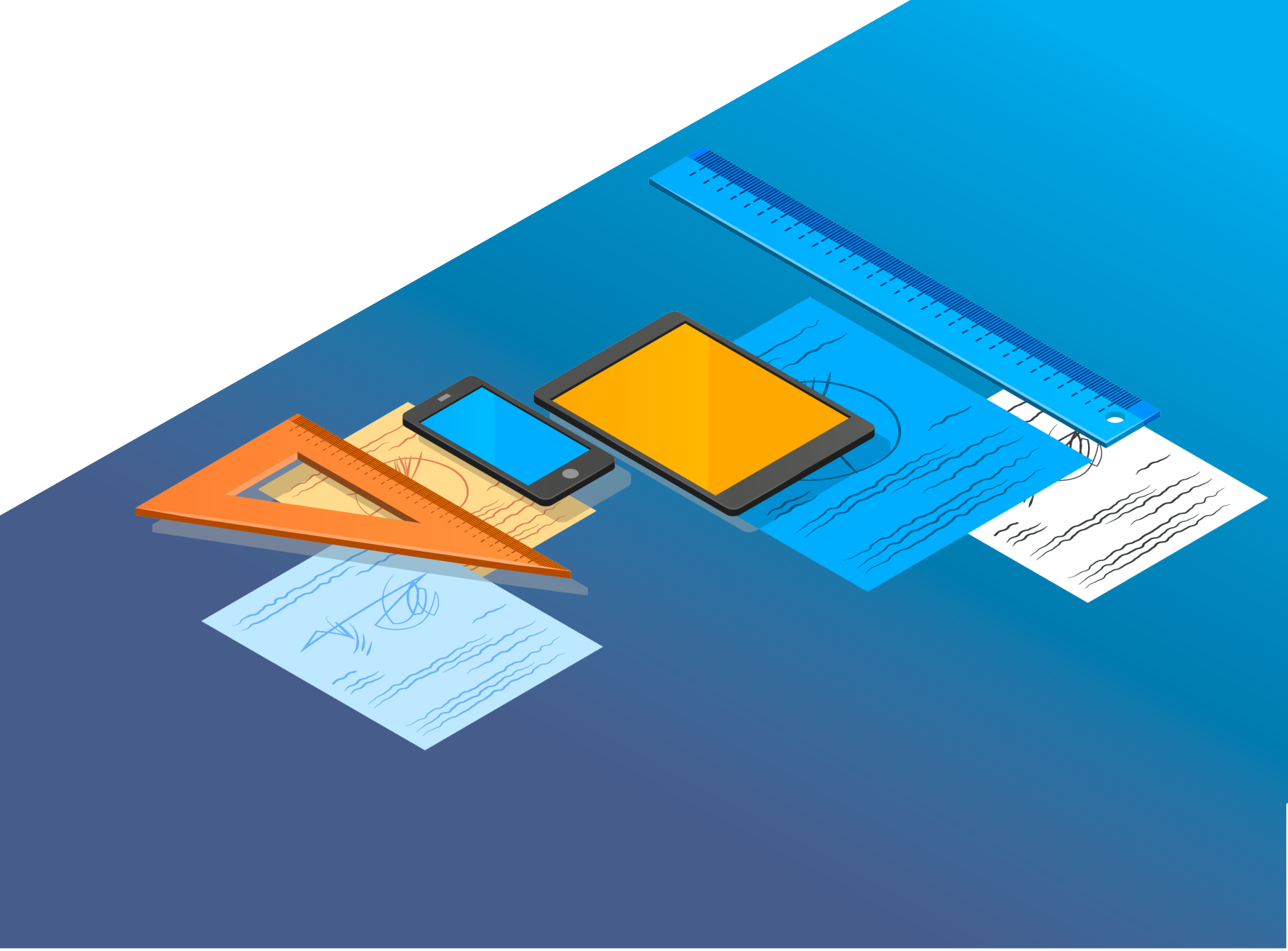
Digital art marketplace Twyla is committed to avoiding making any QA hires. In order to ensure that their product doesn't lose quality as a result, the quality owner (a PM) holds weekly "Test-Fests," where the entire company is invited to spend an hour running tests against the latest features and updates. This increases ownership over quality for everyone from developers and designers to sales and marketers.

"Our Test-Fests help build patience and empathy. We have an organization where half of the team is very technical and driven by optimization, and the other half is very creative and driven by aesthetics. The Test-Fest process allows the two to empathize with one another."

- Douglas Ferguson, VP of Engineering, Twyla

Section Four

Designing Strategic Mobile Test Coverage



The question of what should be tested is a major bottleneck to scaling mobile development. Instead of aiming for more test cases, aim for better test cases. The most important tests are the ones that test areas of your application which, if broken, would have the most significant impact on your business.

Map Test Coverage to Business Value

From the usage scenarios you test to the mobile devices and browsers that you cover, fragmentation makes 100% coverage an impossible goal, especially as your audience scales and diversifies. But by leveraging your user data to determine what browser and OS versions your customers rely on most can help you understand where to focus your testing efforts the most.

Build Better Coverage, Not More Tests

Focus on functionality first; a well-designed suite of 10 critical tests will yield far better returns than a suite of 100 that leaves significant gaps in your quality knowledge. Maintaining and running a test suite that covers the core user flow of your product -- such as logging in, checking out, or uploading files -- will offer greater returns on the time and resources you invest in it than chasing down as many edge cases as possible.

When you're creating test cases -- especially those bound for crowdtesting or automated execution -- try to make them as short and modular as possible. By focusing on testing just one function per test case, you'll receive more deterministic feedback that is easier to triage. Modular tests allows you to reuse common test flows in multiple places. On the Rainforest platform, any test can be written as an "embedded test" that can be reused as needed.

Mobile Coverage Considerations

The nature mobile devices requires acommodating several key testing categories beyond those required for web app testing, including:

OS Coverage

Focus on supporting the most recent OS versions, discontinuing support for older version as they become less important to your user base. Don't forget to include app upgrade testing as part of your OS coverage strategy, as issues can be created as a result of software updates.

Device Coverage

As with OS coverage, focus on device versions that constitute the majority of your user base. Opt to cover more screen resolutions rather than more devices with the same resolution, to get more value out of each test run. For manual testing, don't waste time on basic functional tests. Instead, focus on hardware functionality, such as battery life and cellular settings.

Connectivity Coverage

Network and Wifi connectivity can make or break user experiences, but there are seemingly endless permutations of connectivity scenarios. Including common connectivity snafus -- such as switching between Wifi and LTE and back -- in your QA checklist is critical. Dogfooding provides a scalable way to account for a greater number of connectivity edge cases.

QA Snapshot: Scaling QA with Modular Testing

Temporary staffing platform Jitjatjo has a lean, distributed team with a single QA engineer to support the entire quality process for their mobile application. Every test case written by Jitjatjo is modular, and tests often consist of several nested modular tests. By writing as many reusable tests as possible, Jitjatjo's team is able to spend less time writing and updating test cases, while creating a test suite that scales with their product.

"Rainforest allows us to create a test that we can run over and over. Any time we want to test that flow, we gain the efficiencies back. Any Rainforest test case that we write becomes extremely efficient. We're saving 25-30% of our QA time by leveraging Rainforest."

- Dominic Esposito, Head of Product, Jitjatjo

Section Five

The Automation Endgame

"The fetishization of automated tests as the magic bullet that will allow you to deliver software quickly is wrong. The opposite is the case!"

- Sally Goble, Head of Quality for the Guardian[3]

While testing automation is an excellent tool in a QA team's toolkit, it's not the magic bullet; it's simply not flexible enough to keep up with fast-evolving applications.

The "Total Automation" Myth

Testing automation is often held up as the endgame for QA. Teams mistakenly believe that if they can just crack the automation code, all of their quality problems will be solved. But it's not as simple as that. Even when implemented perfectly, automated tests are brittle and flaky, and require a large amount of maintenance and management.

The most scalable technique for applying automation to your QA strategy is for stable features. However, even stable features may be subject to changes that render their automated tests unusable. The stability of mobile apps for automation is heavily impacted by the fast-changing mobile device market.

In addition to the demand for constant feature upgrades, OS updates, the release of new devices and other "environmental" factors make every mobile app test challenging to automate. While stable automated tests can theoretically be used to quickly and repeatedly execute tests without additional cost, the dynamic nature of the landscape of mobile devices and OS makes stability a continually shifting target.

[3] Sally Goble, "So What Do You Do If You Don't Do Testing?!," PIPELINE Conference 2016

A Dynamic Approach for Better Testing Automation

Automation should be used as one testing tool that can help speed up -- but not entirely replace -- a comprehensive QA execution strategy. Treating automation as the final goal in a test's lifecycle can create a backlog of automated tests that quickly become brittle and unreliable.

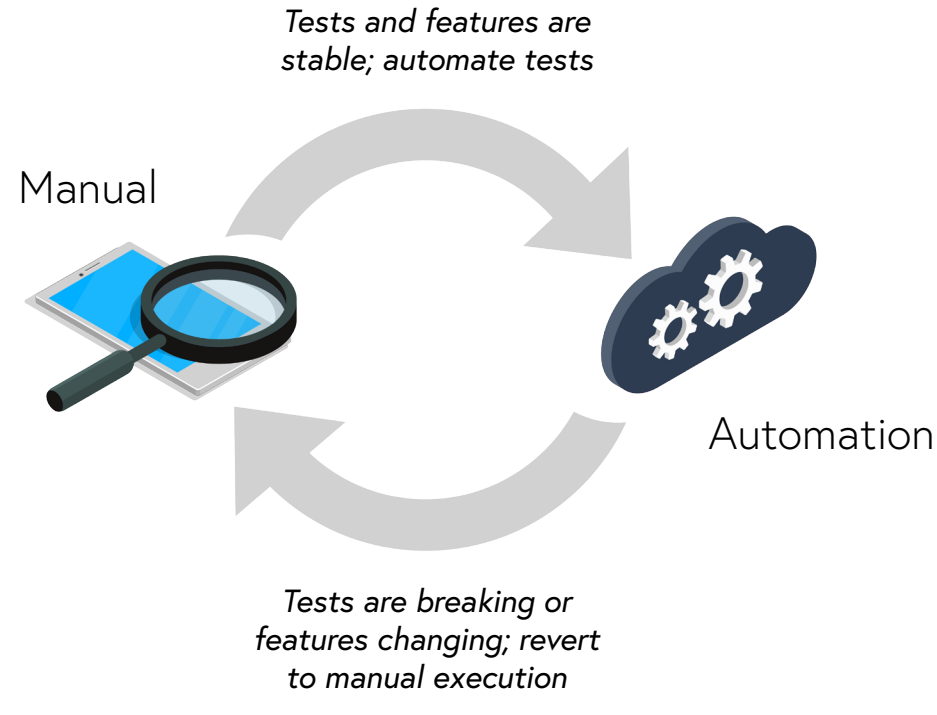
Anticipate Breakage

Automated tests break -- but understanding the most common reasons for breakage will help you spot and get ahead of issues. There are several common breakage scenarios that a recent study[4] suggests developers must look out for. Keep these in mind when considering your automation strategy:

- Element Locators-based Breakage
- Value-based Breakage
- Page Reloading
- User Session Times
- Pop-up Windows
- Timing

Plan for Manual Testing "Rollbacks"

Rather than seeing the road to automation as a one-way street, it should be seen as one type within an ecosystem of testing types for different scenarios. By anticipating that automated tests maybe need to be occasionally "rolled back" to manual testing methods, you can keep your test suite healthier and more reliable overall. Leveraging a rapid-feedback crowdsourced testing solution to execute these tests that are not quite ready for automation helps keeps things running smoothly and quickly.



[4] Mouna Hammoudi; Gregg Rothermel; Paolo Tonella, "Why Do Record/Replay Tests of Web Applications Break?" 2016 IEEE Internation Conference on Software Testing, Verification and Validation (ICST)

Section Six

TL;DR: Key Factors in Scaling Mobile App Testing

Three Critical Takeaways

1. Rethink Your Existing Resources
QA teams offer suffer from using resources inefficiently. Take inventory of the people, tools and processes that you're currently using to do testing, and think of how they can be optimized for better results.

2. Start with a Sprint But Plan for the Marathon
Scaling mobile testing processes requires a proactive, rather than reactive approach. Whenever possible, use testing methods that allow you to reduce the amount of hands-on time you need to spend on testing in the long run.

3. If You Can Automate or Crowdsource It -- Do It
Do whatever you can to avoid doing things manually if you don't have to. The more you can leverage testing accelerators like automation and crowdsourcing, the more you'll be able to scale up mobile QA.

Learn More about Continuous Testing for Mobile

This guide was developed using the principles outlined in the **Continuous Testing Manifesto**. This manifesto outlines the core factors that any development team must follow in order to develop a strategic QA process that keeps pace with agile and continuous development methods.

Read the Continuous Testing Manifesto to learn how to accelerate and scale your QA strategy effectively.

The mission at Rainforest QA is to make quality assurance easy and painless for everyone. Every day thousands of our testers help fast-moving companies figure out what is broken on their sites and applications.

We believe that the future of software development depends on removing roadblocks that slow the development lifecycle. Rainforest helps customers focus on exploiting their competitive advantage by leveraging the kind of elegant tooling that gives small teams the ability to make an outsized impact.

rainforest

2 Embarcadero Center
Promenade Level, Suite R-2308
San Francisco, CA 94111

415-969-6326
www.rainforestqa.com