The era of mobility has arrived. Consumers and corporate IT users are abandoning PCs and laptops in favor of smartphones and tablets, delivering a clear message to enterprise IT: create the new mobile apps needed—now. This leaves software quality teams scrambling to acquire new tools, new skills, and new processes. But in the face of some unique challenges and in the absence of a clear mobile software quality strategy and roadmap, these efforts can leave the business standing at a precipice looking down. Businesses that successfully traverse the precipice will climb to new heights, so this paper shows how to bridge the chasm to compelling, quality mobile apps.

Executive summary

By 2015 mobile application development projects are expected to outnumber PC-based ones by four to one. Application development teams in enterprise IT are gearing up to produce the mobile apps their customers require. But quality management is lagging behind—testing organizations are standing on the precipice of mobility looking for a lifeline. Recent research finds that 65 percent of companies do not have the right tools and over a third are lacking in process and methods. Technology has once again outpaced IT’s ability to adapt to change. But those that do will outpace their competition by offering a superior user experience to their customers and boosting the productivity of their employees.

Ensuring the quality, performance, and security of mobile apps presents new and compelling challenges. Some are driven by the plethora of devices, OSs, and version developers and testers must accommodate, others by the nature of mobile apps, and still others by the Agile development processes required to meet time-to-market demands.

This paper describes the special challenges of mobile testing. We show how mobile apps are different and require a different approach. Then we present a specific set of recommendations to build a bridge from where you are today to a comprehensive and effective mobile test program.

Understand the mobility challenges

Mobile apps are not just another type of application in the mix; they are different in ways that stress existing quality teams, processes, and tools. Let’s look at some of the challenges they bring.

Diverse platforms

There is a huge number of smartphone and tablet models and multiple OSs. This creates a situation where, to comprehensively test an app, one must test across hundreds of devices. This is exacerbated by the rapid pace of release of both new devices and new OS versions. Maintaining a fleet of devices and keeping it up to date is cost prohibitive. As a result, most test only across a sample set of devices, exposing themselves to brand and reputation risk. With bring-your-own-device (BYOD) initiatives, even corporate IT can no longer rely on its ability to narrow the list by enforcing a “standard” set of devices.

Release churn

OS releases for PCs were infrequent, and usually provided backward compatibility. Further, most enterprises only adopted new versions after months or years of testing or when the supplier stopped supporting the version they were on.
The mobile world is different. The pace of release is faster, device vendors modify the OS creating a plethora of one-offs, and backward compatibility is a matter of luck (for example, Apple’s introduction of Flat Design and emphasis on Auto Layout in iOS 7). Users rush to adopt the latest and greatest, which forces you to regression test the existing code for each one and almost guarantees that the code will need to change and therefore be tested again.

**Agile development processes**
To meet the time-to-market demands of mobile users, most mobile apps are developed using Agile and lean development processes. While many development organizations are gaining competency with these methods, mature Agile testing capabilities lag behind. The most common approach is to test mobile apps using a waterfall testing approach. That results in one of two outcomes:

- Testing is sacrificed for the sake of speed. That leads to poorly tested apps, performance issues, and often rollbacks (if possible). It can result in embarrassment in front of customers, the media, and corporate bosses. And it can weaken your brand within user communities.
- Testing slows down the development process and defeats the purpose of Agile development.

Maturing Agile-testing capabilities requires you to change your processes, retrain your staff, and retool your organization. And each of those requires an investment and imposes its own learning curve.

**Complex application integration**
While enterprise applications tend to be organized around business functions—customer relationship management, order processing, and finance—well-designed mobile apps are organized around user tasks. They are composite applications that often span and must be integrated with multiple enterprise applications. In addition, many mobile apps make use of external services like location awareness, mapping, authentication, and other services provided by third parties via APIs and components. For example, a taxi booking service app may authenticate the user using Facebook, and it may locate them using Google and the phone GPS. This presents several challenges:

- You must ensure that your app is robust enough to handle any issues with third-party services.
- You must test for performance when you do not own any of the third-party infrastructure and systems.
- You must ensure the security of your app when your users are simultaneously connected to the corporate network for internal apps and the cellular network for external ones.

**New performance demands**
Mobile applications bring new, unanticipated performance demands, because usage patterns change. Mobility brings thousands of additional users to enterprise applications, and the users interact with the app much more often. People following a sporting event on the web would check the website a few times. Those doing it on a mobile phone check it every few minutes.

And since the carrier networks supporting these connections are not under your control, any performance testing you do—which excludes simulating the network—is incomplete and suspect.

**Outsourcing risk**
Many companies currently developing mobile apps outsource development to third parties, often as a turnkey solution. In that case, you may have no insight into the level of testing that is undertaken. As outsource development contracts typically provide no guarantees or penalties against poor quality or performance, you must perform some level of testing internally to protect your brand. This creates steep learning curves for your people, cost overruns for the organization, and governance challenges.
An approach for mobile testing success

Enabling success with mobile applications requires a new approach to meet the quality challenges they impose. Forward-looking and dynamic quality teams can build a bridge to success using the following approach.

Have a strategy
Ensuring quality, performance, and security for mobile apps cannot be addressed as a bolt-on to existing quality processes. You must integrate mobile testing into your application lifecycle management and testing practices. This means looking at all elements of a strategy—people, process, and tooling.

Start with what you have
Perform an internal audit and assessment of your current software testing capabilities, resources, and infrastructure to provide a baseline for improvements. Then consider anticipated applications, environments, platforms, budget, and time-to-market needs. This will help you determine the desired future state and allow you to develop a long-term plan supplemented with short-term solutions.

Develop a long-term plan
The objective of your long-term solution is make mobile testing as agile and lean as possible without sacrificing quality, security, and performance. The mobile market is fluid; coding languages, platforms, costs, and technology change rapidly. So invest in processes and technology that are flexible and have a three-year roadmap that parallels your own.

Flexibility should be the cornerstone of your long-term plan, and you should avoid becoming boxed into a single solution. For example, choosing software as a service (SaaS) as the deployment option for technology can save your company money upfront, and in many cases is an excellent choice. However, make sure that the data is portable so you can deploy the technology in house should you decide to later. Also, make sure the technology is compatible with your current systems, so you will not have to maintain multiple skill sets. The same is true of your testing environments if you’re looking at cloud-based solutions.

Provide short-term capabilities
Your long-term plan will take time to achieve—time you may not have. So identify a short-term solution that can jump-start your mobile testing efforts while transitioning into your long-term configuration.

A hybrid solution such as testing as a service or SaaS could aid you in developing a testing strategy by osmosis. Rather than learning it all by yourself, you can engage an external vendor with the experience and the skills you need. This will allow your internal team to assimilate the new strategies, technologies, and processes, and it can ease the burden of a steep learning curve.

Don’t approach it as a hands-off, outsourcing strategy but rather one that provides a bridge to the future. As you ramp up the number and frequency of releases, you gradually insource critical capabilities until you have acquired the skills and competencies you need. Then you can outsource to add value to your efforts, rather than substitute for them.

Enable organizational learning
As you gain competency, make sure you provide a way to leverage learning across all teams, share best practices, and enforce standards. An internal Center of Excellence (CoE)—either as a dedicated team or a virtual organization—focused on mobile application quality can help control critical aspects of mobile software quality management: technology, processes, standards and methods, and knowledge retention. You can then combine internal capabilities with external value-add technology and labor solutions to meet special needs.

Get the right resources on board
Staffing mobile test projects requires expanding teams and getting the right skills. Because hiring and training staff is expensive and time consuming, you must balance hiring with the time-to-market demands of the project. As mentioned above, third-party services can play an important role in staffing.
The table below identifies several staffing options and the situations for which they may be the best fit.

**Table 1. Staffing options**

<table>
<thead>
<tr>
<th>Staffing option</th>
<th>Best application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrain current staff</td>
<td>• Small mobile projects with mature and lean testing strategy</td>
</tr>
<tr>
<td></td>
<td>• Mobile apps linked to legacy apps and using current APIs</td>
</tr>
<tr>
<td></td>
<td>• Highly technical subject matter experts (SMEs)</td>
</tr>
<tr>
<td></td>
<td>• Single platform mobile app, 3–4 annual releases</td>
</tr>
<tr>
<td>Expand current group</td>
<td>• Small to medium mobile projects with some type of lean testing strategy already in place</td>
</tr>
<tr>
<td></td>
<td>• Highest value for cross-training of personnel (application SME and mobility SME)</td>
</tr>
<tr>
<td></td>
<td>• Best model for groups that respond better to mentoring than external training</td>
</tr>
<tr>
<td></td>
<td>• Internal facing mobile apps, or more than one mobile platform or OS</td>
</tr>
<tr>
<td></td>
<td>• In most cases has the highest upfront cost</td>
</tr>
<tr>
<td>Create new group</td>
<td>• Large mobile projects that are linked directly to lean development processes</td>
</tr>
<tr>
<td></td>
<td>• Externally facing apps or supporting multiple mobile platforms and OSs</td>
</tr>
<tr>
<td></td>
<td>• New applications being developed independent of existing applications and platforms</td>
</tr>
<tr>
<td>Testing as a service (TaaS)</td>
<td>• Large mobile projects with aggressive timelines and/or new development strategies</td>
</tr>
<tr>
<td></td>
<td>• Externally facing applications, multiple mobile platforms, or OSs, monthly releases</td>
</tr>
<tr>
<td></td>
<td>• Apps developed by lines of business (LOB) independent of IT</td>
</tr>
<tr>
<td>Hybrid—CoE and staff augmentation</td>
<td>• Highest potential for ROI in sprint/releases beyond the first six months to a year</td>
</tr>
<tr>
<td></td>
<td>• Corporate initiative to incorporate internal and external mobile applications into the corporate structure or committed to COE initiative</td>
</tr>
<tr>
<td></td>
<td>• Works best when companies have aggressive timelines for mobile development but are committed to supporting and developing mobile apps as long-term solution</td>
</tr>
<tr>
<td></td>
<td>• Works best when short-term solutions involve using TaaS or consultants while current staff are being trained and infrastructure being built out</td>
</tr>
</tbody>
</table>

**Apply automation**

The upfront investment in time and resources required to automate test cases has slowed adoption of automated testing, even though the benefits are clear. In the past, waterfall world, you could survive with spotty adoption and mediocre capabilities. But in the mobile, Agile world, platform diversity and release churn make automation a must. Even a little automation, especially when developing composite apps with external APIs, can drive down costs and dramatically improve quality.

Fortunately, technology has advanced to enable you to shorten the time to value. Capabilities such as keyword-driven tests, mobile test accelerators, and integration with real device testing tools accelerate the development of automated tests. When selecting automation technology, consider the following:

• Have a technology blueprint. Avoid ending up with a large portfolio of niche, point solutions that you need to integrate.

• Opt for vendors that bring not just products and services, but also intellectual property.

• Select products that are integrated with what you have or are an extension of it.

• Favor tools that will leverage existing skills. The cost of retraining your staff can far outweigh your savings from buying a cheaper tool.

• If you are developing apps with third-party APIs or components, ensure the tool can support testing them via service virtualization features or something similar.

• If you opt for SaaS tools, avoid vendor lock-in. Pricing models may change, or newer technology may emerge. Make sure your test cases and other data are portable, so you can migrate to a different tool if needed.
Invest in a test automation framework that provides prebuild libraries for reusable functions like error and exception handling, data management, and advanced features such as prebuild keyword libraries for common functions. This will significantly reduce the cost of automation and improve the quality and reliability of test scripts.

Take advantage of third-party test platforms
To test mobile apps, you need a suitable test environment. Simulators are useful for development, but you must test on real devices. If you are developing apps for your employees and can restrict the choice of devices, you might be able to acquire and maintain the test devices you need. But in most cases, the myriad of different devices, OSs, releases, and connectivity packages makes that impractical or cost prohibitive.

Cloud-based platform-as-a-service providers offer on-demand, pay-as-you-go services that let you access a rich array of devices without the overhead of owning and managing them. Make sure that your automation platform integrates with the provider you choose.

Integrate large-scale testing into Agile development processes
Many testing organizations will attempt to retrofit current testing strategies and processes to a high-velocity, lean development process. Or they believe they can rely on unit testing. This type of afterthought, coupled with high-velocity development, poses one of the greatest risks to the security and performance of the app.

Most Agile processes focus on communications between the business user and developer. When designing your mobile testing processes for Agile development, look at concepts like continuous integration and testing, and “left shifting” the test design stage.

Continuous integration and testing
Continuous integration is the straightest line between quality and quantity. It improves software quality and reduces the time to deliver it by applying quality control in smaller pieces of effort and with more frequency. It automatically initiates tests to validate the build as soon as code is checked in. This necessitates test automation, and it provides far greater coverage than manual testing, because automated testing can execute 24x7.

However, the implementation of continuous integration for mobile demands time, even for an experienced, skilled staff. We often recommend that continuous integration be established after a few sprints—once the Agile teams become intimate with the mobile application environment.

Test-driven development
Even with the most agile groups, test case design often follows development. As a result, test automation lags development by several sprint iterations, so teams rely on manual testing, which creates an increasingly larger gap in test coverage as you struggle to catch up.

Test-driven development “shifts left” testing strategy. By combining user stories with test design, testing and development can stay synchronized. Test automation tools with features like keyword testing allow you to create test cases before code is complete, so as soon as executable code is available, you can test it. In most cases, automated testing for new functionality can be executed within the same sprint in which it was developed. Using source code management (SCM) systems to trigger these automated tests can give development team’s overnight feedback on new and existing functionality.
Figure 1. Left shift testing with test-driven development

Using continuous integration or the shift-left testing approach, coupled with automated API testing to continually test the backend systems, will blur the line between unit, integration, system/functional/regression testing, and even security testing. So performance and other non-functional testing remains only one iteration in arrears instead of many sprints behind development.

Focus on the mobile difference
Besides using the tried-and-true methods of verification and validation, you can break mobile testing down into three key areas excluding unit and integration testing.

Functional
Functional testing, which includes both new functionality and regression testing, is not unique to mobile. But it verifies that the application has met all functional requirements, and in many cases, it is the first time the application will be tested against a fully integrated system. For these reasons, functional testing is an important step before moving on to security and performance testing. Full regression testing is almost always required when dealing with so many devices and operating systems, and the only way to achieve that level of coverage is to use automated testing tools.

Security
While functional testing is a given, mobility requires special emphasis on non-functional testing—security and performance.

Security should have the highest visibility within the software development lifecycle, especially when building composite applications. We recommend you do at least one static security scan with each compile of new code and follow up with at least two dynamic scans before releasing the application to production.

Performance
Depending on your mobile footprint, you can expect four to eight times more activity on your back-end infrastructure than from typical Web applications. For example, in the past, travelers would check their email when they had access to their laptop and a network. Now, they do so every few minutes. This is exacerbated by users’ lack of tolerance for anything slower than five seconds, resulting in endless refreshes. So performance testing is even more important than it is for traditional apps.

Because mobile devices themselves impose more latency than a PC, we recommend a hybrid approach to performance testing that uses simulators in conjunction with a few key mobile devices. Adding a WAN emulator to create variations in the communications layer helps obtain true end-to-end transaction times.
Choose the right quality management partner

More organizations use HP Application Lifecycle Management (ALM) software and services to deliver quality applications than any other solution. That’s because we understand quality, and we know how to help our customers achieve it.

We have enhanced the HP ALM suite of tools to meet the special challenges of developing and delivering mobile applications for the enterprise. What’s more, we have augmented our software products with our services-delivered HP Mobile Test Accelerator framework—preloaded and preconfigured components that implement many of the concepts presented in this paper and help test organizations get off the ground quickly with mobile application testing. When combined with our comprehensive suite of planning, consulting, and testing services, we offer the full set of products and services you need for success with mobile application testing.

Through our products and services we’re helping businesses worldwide meet the challenges of mobility and deliver applications that change the way their employees and customers work.

Learn more at hp.com/go/mobiletesting