TATA CONSULTANCY SERVICES
Experience certainty.

TATA

White Paper

Mobile Application Testing

Device and platform diversity, short release cycles, lack of mature testing tools and the variety of network connectivity options result in frequent cost overruns and missed deadlines in today's mobile application testing environment.

A comprehensive mobile testing strategy that includes device and network infrastructure, optimized selection of target devices, and an effective combination of manual and automated testing tools to cover both functional and non-functional testing is essential for getting your mobile applications to market on time and within budget.

This paper will discuss various elements of an effective mobile testing strategy and suggest ways to optimize testing of mobile applications.

About the Author

Tushar Pradhan

Education

Product Management – Haas School of Business, University Of Berkeley

B.E. (Computer Science) – VJTI, Mumbai University

Summary of Experience

Tushar Pradhan is product manager for Hy5 Test™ - an automated solution for testing mobile applications. He has more than 15 years of experience in the mobile space in product management and engineering roles, and has led a variety of mobile products and services including mobile test tools, cloud services for developers and device management solutions at Adobe, DeviceAnywhere and Innopath. His primary interests lie in mobile development tools and ecosystems especially for Android and iOS.

Table of Contents

Executive Summary

The number and variety of consumer and enterprise mobile applications has grown exponentially over the last few years. Organizations need to ensure that every application meets a high quality bar in order to prevent revenue loss, lost productivity and damage to brand reputation.

Testing mobile applications is different and more complex than testing traditional desktop and web applications. Mobile applications need to be tested on a variety of software platforms and versions, on diverse hardware and form factors, and under different network connectivity conditions. Moreover, the rapid pace of mobile OS updates, the frequent introduction of new devices and the customer expectation of quick upgrades require additional test cycles.

A comprehensive mobile application testing strategy is essential for getting your applications to market on time and within budget. Key elements to consider for effectively testing applications are –

- Target Device Selection – Create an optimal mix of simulator testing and physical device testing on different models to maximize test coverage.
- Test Automation – Select an effective test automation tool and maximize the use of automation to reduce the cost of regression testing.
- Network Environment – Consider testing primarily on Wi-Fi networks and using network simulation tools to simulate cellular connectivity and various network conditions.
- Types of Testing – Consider different types of testing required (functional, performance, security, and compliance)

The challenge of mobile application testing can be effectively addressed by a test strategy that combines these elements with traditional best practices and processes for testing.

The Challenge of Testing Applications

Smartphone applications have experienced explosive growth since 2007. The combined number of applications in the Apple App Store and Google Play already exceeds a billion. Additionally, every major enterprise is rapidly adding applications for internal use.

With so many applications competing for users' attention, meeting a high quality bar is essential for the success of an app. Poor-quality applications not only hamper user adoption but also can cause revenue loss and irreparable brand damage. Defective enterprise applications can lead to lost productivity.

Why Mobile App Testing is Different

Testing mobile applications is more complex and time consuming compared to traditional desktop and web applications.

The majority of desktop applications need to be tested on a single dominant platform – Windows. The lack of a similar dominant platform for mobile apps results in many apps being developed for and tested on Android, iOS and sometimes even more platforms.

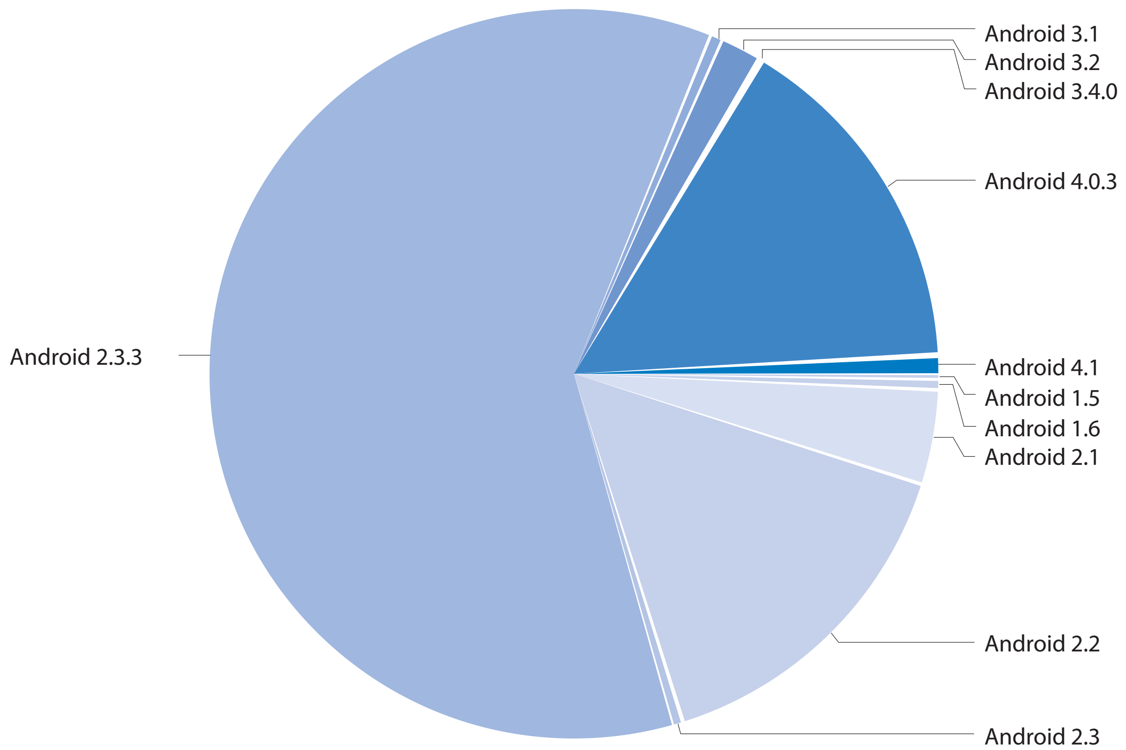

Figure 1 - Android OS Version Fragmentation

The slow pace of OS updates on Android devices (see figure) and the resulting OS fragmentation results in the need to test apps on various versions of Android.

Unlike the desktop world, where PCs are established as standardized reference hardware, the wide variety of device form factors (e.g. phones and tablets of various screen size) adds another layer of complexity in testing mobile apps. Device diversity is an especially acute problem for Android devices – even the official Android device gallery[1] includes over 60 devices of various screen sizes, resolutions and form factors.

The ease of upgrading apps over the air combined with increased user expectations about quicker releases (both for bug fixes and new features) result in frequent application releases. Adding multiple major and minor OS updates on top of this, test teams are continuously tasked with testing new app features or recertifying the app against a new OS version.

Mobile apps operate in a unique environment where application behavior can be affected by changes in network conditions (bandwidth change, dropped connections), alerts and notifications, as well as touch screen responsiveness. This unique environment requires additional testing to ensure acceptable app behavior in real world conditions.

[1]http://www.android.com/devices/

All these factors are responsible for the high cost of testing mobile applications. At TCS, we regularly hear from customers that are trying to reduce the cost of mobile app testing which sometimes exceed development cost.

Dimensions of Testing Mobile Apps

In order to understand the complexity of testing mobile apps, it is important to understand various aspects of an application that need to be tested. Some of these aspects are specific to mobile applications while others are applicable for testing any type of software.

The specific types of testing required for each application depends on various factors such as:

- The type of application (banking, gaming, social, or business)
- Target audience type (consumer, enterprise) and volume
- Distribution channel (e.g. Apple App Store, Google Play, direct distribution)

Aspect	**Areas / Types of testing**
Functionality	User Interaction Testing Transaction testing
Performance	UI responsiveness Transaction completion time(s) Peak load performance Longevity
Network	Network type (Wi-Fi, 2G, 3G, 4G) Impact of Connectivity Issues
Security	Data Retention on device Transmission Security
Compatibility	Mobile Platform Compatibility (e.g. iOS 6, iOS 5.1.1, iOS 5.1.1) Device Model Compatibility Backward compatibility (with previous app version)
Conformance	Marketplace guidelines compliance (e.g. Apple App Store policies) Enterprise policy compliance (e.g. prohibited content)
Usability	User Experience
Installation and Provisioning	Installation process Un-installation process User provisioning and de-provisioning

Application Lifecycle and Testing

Mobile applications are upgraded frequently for a variety of reasons –

- Mobile users expect rapid feature upgrades and bug fixes
- As more and more developers use agile development methodologies, frequent releases with incremental feature updates are becoming common
- Applications are updated to take advantages of new versions of mobile platforms (such as iOS or Android)

Each application upgrade warrants an additional test cycle. Moreover, additional test cycles are needed when a new version of a mobile platform (such as iOS or Android) is released to ensure application compatibility. Similarly, additional test cycles are needed when a new high profile device is introduced in the market.

The scope of testing for each test cycle depends on the underlying changes. Some minor changes may require testing only key application functionality (often called 'smoke test' or 'sanity test') while other changes may require full regression testing and testing new features.

Mobile Test Tools Landscape

Effective testing tools are essential for addressing the challenges and complexities of mobile application testing.

Remote Access Solutions for Manual Testing

In the pre-smartphone era, it was important to test applications on a large number of devices on major operator networks. Vendors addressed this need by providing a 'remote access' (aka 'device cloud') solution that allowed testers to access devices hosted in various data centers over the Internet through a browser or client application. These services were eventually extended to include smartphones. Users typically pay for these services based on the duration of device usage.

Today's smartphone market is dominated by iOS and Android. Typical applications need to be tested on 5-7 iOS devices (iPhones and iPads with a mix of latest iOS versions) and 10-15 Android devices (most popular Android phones and tablets representing various Android versions, screen sizes and resolutions). Moreover, the majority of testing can be carried out on Wi-Fi networks rather than carrier's cellular networks. Tools such as TCS WANem even allow simulation of cellular networks (including bandwidth variations and packet drops) over a Wi-Fi network.

The average hourly usage cost of remote access solutions is $25-$30. With average iOS and Android device costs of $500-$600, remote access solutions may not be cost effective compared to testing with live in-house devices.

Automation Tools

Automation tools for mobile application testing use one of two major technologies:

Object based tools (such as TCS Hy5 Test™ and Jamo Solutions) achieve automation by mapping elements on the device screen into objects and manipulating them. This approach is independent of screen size or resolution and provides a high degree of script reusability. This can be especially important for Android devices where variations in screen sizes and resolutions are widespread. Some object-based tools (e.g. Jamo Solutions) require changes to application source code while others (e.g. TCS Hy5 Test™) do not require any source code changes.

Image based or bitmap based tools (such as Perfecto mobile and DeviceAnywhere) create automation scripts based on screen coordinates of elements. For example, tapping a button on the screen is achieved by tapping the coordinates (e.g. x=35, y=40). While this approach is agnostic to the type of application (native, web, hybrid) and provides useful image matching capabilities, it typically requires jailbroken or rooted[2]devices and scripts to be re-written for each device with a different screen size or resolution (due to change in coordinates). Jailbroken devices are typically incompatible with enterprise MDM security policies and hence tools that rely on jailbroken devices may not be suitable for testing enterprise apps.

Some of the current automation tools are characterized by very high initial investment and/or ineffective ROI.

Unfortunately, none of the current tools provide a cross-platform, cost effective test automation solution that combines the benefit of all available automation technologies. TCS is investing heavily in its Hy5 Test™ test automation tool to meet this gap.

Mobile Application Testing Strategy

Various elements of an effective mobile application testing strategy are considered in this section. Recommendations in this section are based on TCS' extensive experience in developing and testing scores of mobile applications.

Selecting Target Devices

Target devices for testing a mobile application should balance the need to use a representative sample of the expected device population with the need to optimize duration and cost of testing.

[2]'Jailbreak' (iOS) or 'rooting' ('Android') is the process of hacking into a device to install and execute applications prohibited by OS / device vendor.

Simulators vs. Physical Devices

Use of device simulators can be highly effective in the early stages of product development when features are under development. Simulators are useful to familiarize the test team with various application features.

Figure 2 illustrates various conditions under which testing with simulators and real devices can be effective. Simulators can be effectively used for testing basic application functionality. Testing on physical devices is essential to understanding application behavior such as touch response and user experience. Stable, defect-free (based on simulator testing) features can then be tested on physical devices. Device usage can be optimized by distributing test cases across various types of devices.

Note that this strategy requires that the test tools being used support testing with simulators as well as devices.

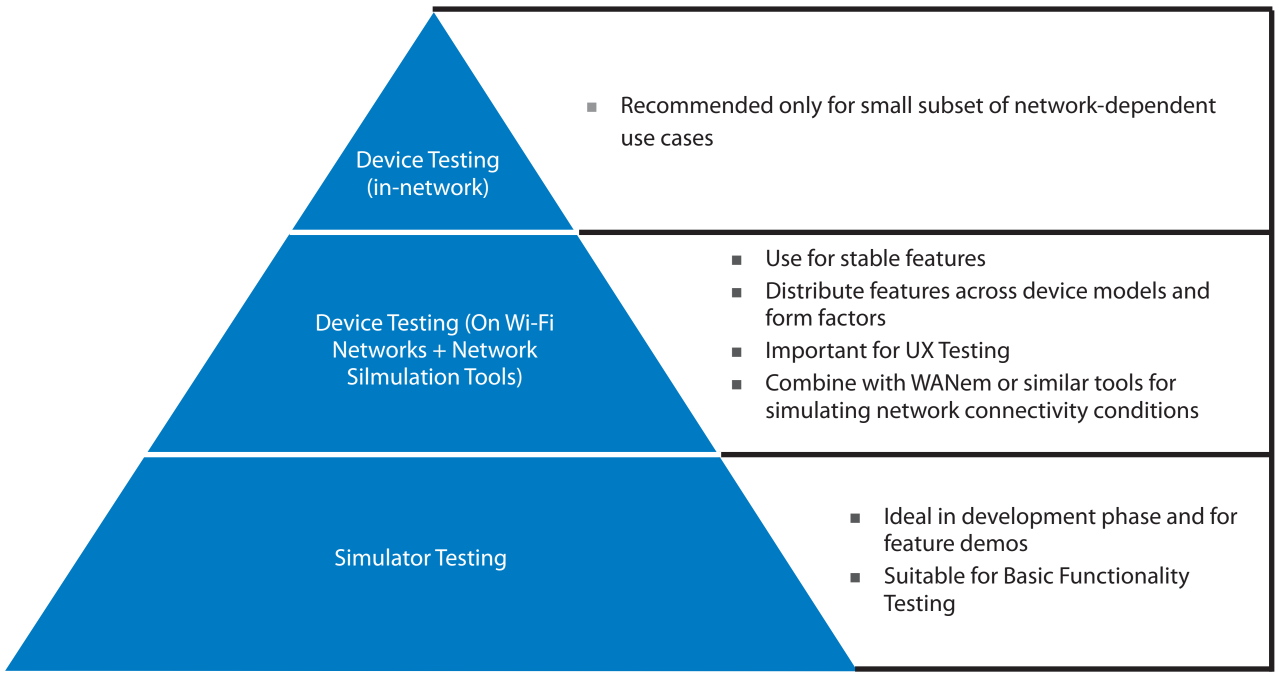
Device Testing (in-network)
Recommended only for small subset of network-dependent use cases
Device Testing (On Wi-Fi Networks + Network Silmulation Tools)
Use for stable features
Distribute features across device models and form factors
Important for UX Testing
Combine with WANem or similar tools for simulating network connectivity conditions
Simulator Testing
Ideal in development phase and for feature demos
Suitable for Basic Functionality Testing

Figure 2- Conditions for Testing with Devices / Simulators and Different Connectivity Options

Device Model Selection

Key factors in deciding specific device models for testing are –

- OS Version – Applications should be tested on all major OS versions in the current installed base. For Android devices, OS distribution of various Android versions provided by Google at http://developer.android.com/about/dashboards/index.html can be used to determine the mix of Android versions to be targeted.

For iOS devices, previous experience shows that 90% of iOS devices can be expected to be upgraded to the last major version (see http://david-smith.org/iosversionstats/). For example, when the latest available version is 5.1.1, over 90% of devices can be expected to be on version 5.0 or later.

- Form Factor – Applications may behave differently (especially in terms of user experience) on smartphones and tablets. If the test application supports both smartphones and tablets, it should be tested on both form factors
- Display Density – The screen size and screen resolution of a device (aka display density) can also affect application user experience (e.g. look and feel of UI, rendering of web pages). Device models selected for testing should include a mix of different display densities.

A variety of other factors (keyboard, CPU type, memory) can affect specialized applications that rely on specific device features and should be taken into account when necessary.

Selecting specific iOS device models can be relatively easy due to the small number of device models available. A commonly used and effective method for selecting Android test devices relies on selecting the most popular devices based on market data and then adjusting for the above parameters. TCS maintains a recommended list of target devices for testing that is periodically updated based on market changes.

When testing enterprise applications, a user survey or device model data from internal systems can be a good source of information to determine the mix of target devices for testing.

Connectivity Options

Almost all applications rely on network connectivity to provide useful functionality. In test environments, Wi-Fi networks can be easy to set up and can be cost effective, unlike cellular connectivity that can be both unreliable and expensive.

Network simulation tools such as TCS WANem provide a cost effective and easy to use option to simulate cellular networks using a WAN/Wi-Fi connection. These tools can be used to simulate various network speed/bandwidth options (2G, 3G, and 4G), connectivity issues, and bandwidth variations.

On-location testing on specific cellular networks should be considered only for specialized applications that rely on carrier features such as free usage allowance or two way SMS.

Manual vs. Automated Testing

Automated testing provides a mechanism to consistently repeat a test procedure and verify application results. It can be effective both for regression testing as well during development (to ensure that new features are not resulting in unintended changes to existing features).

However, automation requires a significant initial investment (in a test tool as well as scripting) and the ROI is realized when the same automated test is executed multiple times with negligible incremental cost. Therefore, it is important to judiciously select candidate test cases for automation.

The following guidelines are recommended for automation of mobile application testing –

- Features that are expected to change in the near future (especially in terms of UI flow) should be avoided for automation.
- Some test cases may be extremely expensive to automate due to technical challenges or unique use cases (e.g. handling real time data from multiple sources), negating the benefits of automation. Such test cases should be avoided for automation.
- Every possible test case should be automated unless it meets the above criteria for avoiding automation.

To achieve the maximum benefit from automation, all automatable test cases from a release should be automated before the next release.

Key use cases for automation in mobile application testing are –

- Verifying application compatibility when a new iOS, Android (or other applicable) OS version is released
- Verifying backward compatibility when the application is upgraded

Selecting a Test Automation Tool

As described in section 1, the lack of an ideal test automation tool makes the selection of a test automation tool a complex decision. The following table lists key criteria that should be considered when selecting a test automation tool.

Criterion	Areas / Types of testing
Multi-Platform Support	Consider your current and future target platforms and ensure that the tool can support them.
Script Reusability	Object based tools such as TCS Hy5 Test™ provide a high degree of script reuse. Image based tools generally have lower script re-use resulting in higher cost.
Jailbreak Requirement	If the tool uses jailbroken or rooted devices, it may not support latest OS versions and may be incompatible with MDM policies required for enterprise apps.
Source Code Changes	Sharing source code may not be always possible and changing the source code increases the risk of missing some defects.
Lead Time for New OS version / Device	How soon can the tool support new iOS/Android/other OS versions?

Criterion	Areas / Types of testing
Test Workflow	Does the tool provide an easy to use workflow?
Results and Reports	Does the tool archive results and provide reports?
Integration Capabilities	Can the tool integrate/interoperate with other test management/bug tracking/build management systems?
Deployment	What deployments options are available? Can it be deployed in-house/in a 'cloud model'?
Price	How much does it cost?
Service and Support	What services (e.g. script development) and support are offered?

Conclusion

Despite the challenges in mobile application testing, careful selection of target devices, connectivity options, and tools that maximize automation can ensure a cost effective mobile testing process.

An optimal selection of target devices and using a mix of simulators and physical devices can maximize test coverage without the need to test every feature on each device. The use of Wi-Fi networks for the majority of testing in combination with network simulation tools can reduce the cost and complexity of testing on various cellular networks.

Maximizing automation is an effective way of expediting the testing process and reducing long term testing costs. Factors such as support for applicable mobile platforms, script reusability and total cost of ownership should be taken into account when selecting automation tools.

Combining the solutions to mobile specific aspects of application testing with traditional best practices and testing processes can effectively address the challenges of mobile application testing.

References

[1] Android Version Distribution: http://developer.android.com/about/dashboards/index.html

[2] iOS Version Distribution: http://david-smith.org/iosversionstats/

TATA CONSULTANCY SERVICES

Experience certainty.

About TCS Mobility

TCS Mobility delivers best in class mobility services and solutions with complete mobility lifecycle consultation and development service along with customization on top of pre-built solutions to provide the best in class benefits to our customers.

Our passion for providing the very best and comprehensive mobility services and solutions to our customers is realized through our deep expertise in mobility gained through the experience of a strong team that has a career long experience in mobile technologies and a dedicated mobility user experience design team that is committed to leveraging the unique native capabilities of each device platform.

Contact

For more information about TCS' Mobility contact **mobility.solutions@tcs.com**

Subscribe to TCS White Papers

TCS.com RSS: http://www.tcs.com/rss_feeds/Pages/feed.aspx?f=w
Feedburner: http://feeds2.feedburner.com/tcswhitepapers

About Tata Consultancy Services Ltd (TCS)

Tata Consultancy Services is an IT services, consulting and business solutions organization that delivers real results to global business, ensuring a level of certainty no other firm can match. TCS offers a consulting-led, integrated portfolio of IT and IT-enabled infrastructure, engineering and assurance services. This is delivered through its unique Global Network Delivery Model™, recognized as the benchmark of excellence in software development. A part of the Tata Group, India's largest industrial conglomerate, TCS has a global footprint and is listed on the National Stock Exchange and Bombay Stock Exchange in India.

For more information, visit us at **www.tcs.com**

IT Services
Business Solutions
Outsourcing