Continuous Mobile Application Testing
With SOASTA TouchTest™
Table of Contents

Executive Summary................................................................. 3

1. The Mobile App Revolution and the Testing Challenge ...................... 4

2. Other Mobile Testing Products Cannot Meet the Challenge .................. 5
   2.1 Problem – Jailbreaking of iOS devices is mandatory .................. 5
   2.2 Problem – Tethering is required at all times ......................... 6
   2.3 Problem – Optical recognition is fragile for functional validation .... 6
   2.4 Problem – Not viable for continuous integration testing ............. 6

3. SOASTA’s Solution – TouchTest............................................ 6
   3.1 No jailbreaking required for iOS devices ............................... 6
   3.2 Testing unleashed – no tethering required ............................. 6
   3.3 Native level validations ....................................................... 7
   3.4 Full support for continuous integration .................................. 7

4. The Blueprint for Mobile Application Automation ............................. 8
   4.1 TouchTest CI Architecture .................................................. 9
   4.2 The CloudTest Methodology ............................................... 10

Conclusion................................................................................... 11

About SOASTA, Inc. ..................................................................... 11
Executive Summary

This white paper discusses challenges specific to testing mobile applications, how SOASTA TouchTest addresses those problems and offers a blueprint for how continuous mobile app testing should be executed in the SDLC.

Mobile is today’s fastest growing technology sector. In the week between Christmas 2011 and New Year’s Day 2012 more than one billion apps were downloaded. Nearly every company involved in technology has a strategy to release apps or web sites optimized for mobile devices. The surge in mobile app development has brought a new set of challenges that engineering teams need to overcome in order to keep up with this rapid rate of growth.

The sudden addition of mobile devices and apps into the SDLC has left many teams with only one way to test their applications: “fingers and eyeballs”. This means that manual testing is being done all throughout the pre- and post-release lifecycle. Manual testing is problematic for many reasons: it drastically slows down the development process, leaves a huge margin for error, and ultimately lowers confidence in a team’s ability to release quality software in a short amount of time.

The few testing products that have been introduced into the mobile automation space are consistently failing in the areas of speed, accuracy and continuous integration across the entire lifecycle. SOASTA TouchTest™ is an end-to-end mobile testing solution that allows teams to release quickly and with confidence that their apps will deliver a great customer experience. This white paper discusses challenges specific to testing mobile applications, how TouchTest addresses those problems and offers a blueprint for how continuous mobile app testing should be executed in the SDLC.
Apple reports that over 25 billion apps have been downloaded from the App Store since its release. Google reports that Android app downloads have broken the 10 billion mark and are growing at a rate of over one billion per month.

1. The Mobile App Revolution and the Testing Challenge

With the explosion of mobile phone and tablet sales in 2011 and 2012, it’s become exceedingly clear that the mobile revolution is upon us. Experts are projecting that tablet sales will surpass PC sales within the next few years. This proliferation of mobile devices and applications has hit companies at an interesting time with regards to software engineering practices.

Since just before the mobile revolution began, companies have begun to focus on adopting agile software practices across the lifecycle. Teams of all sizes are striving for continuous integration cycles that build, deploy and test on more rapid intervals than they’ve ever done before. Development teams that released once per quarter or once per month are trying to execute full build and test cycles weekly, or even multiple times a day. Currently, few organizations are happy with their continuous integration strategy. Yet just as companies have embarked on the journey for best-in-class development and test practices, they’ve been presented with the additional challenges introduced by mobile development.

Enterprises report that testing is still the most problematic phase in the software development lifecycle. This stems from a long history of software testing dominance by a few testing tools that became popular during the client-server era; these include Hewlett Packard QTP (for functional), LoadRunner (for performance) and Micro Focus SilkTest and SilkPerformer. Since HTTP/HTTPS were add-on protocols for these products many years after their introduction, they’ve struggled to keep up with today’s SDLC. These products require writing code in complex, outdated or irrelevant languages in order to test applications, typically C, BDL (similar to Pascal) or VBScript.

Testing has historically been deferred to the end of the SDLC for execution. This was because teams needed a finished build of the software so they could start writing the complex test code over days and weeks, then execute the tests for a pass or fail. With continuous integration servers doing rapid build and deploy, most agree that testing should be part of this cycle. However, the traditional tools haven’t been able to keep up in the new age of software development, and therefore development practices that strive to be agile still do not have an agile lifecycle, just agile development, since testing hasn’t been able to play an effective role. Agile testing is a mandatory component of the entire ecosystem.

SOASTA’s award-winning CloudTest® platform is the first truly viable solution for continuous integration testing of web and mobile applications. The platform allows teams to quickly create, modify and execute tests. CloudTest was born in the age of online applications and is web-based, designed for use in modern agile development and testing organizations. CloudTest has plugin integration with Jenkins and can integrate as an ANT task in any CI server, delivering test results in the standard JUnitXML format. With CloudTest integrated into their continuous integration server, companies can release multiple builds per day with unit, functional and performance tests all conducted in an automated fashion. This is how SOASTA has enabled companies to achieve very quick functional and performance test results with online, web-based apps. The mobile revolution needs to be equally agile, and TouchTest™ is SOASTA’s answer to the challenge.

(1) http://www.wired.com/gadgetlab/2012/03/app-store-25-billion/
(2) http://techcrunch.com/2011/12/06/android-market-10-billion-apps-served-so-far-and-another-1-billion-each-month/
SOASTA's CloudTest Methodology pioneered the practice of testing web applications at scale in production environments. The methodology has been successfully adopted by six of the top 10 online retailers in the US.

Best-in-class results are not just about products though. SOASTA's CloudTest Methodology pioneered the practice of testing web applications at scale in production environments. The methodology has been successfully adopted by six of the top 10 online retailers in the US, and by many of SOASTA's more than 400 customers. The CloudTest Methodology is a framework of lightweight processes that call for frequent building and testing of applications in lab and production environments, along with best practices for doing so in live environments. At its core, the methodology exists to create the basis for lab-to-launch quality and performance. The CloudTest Methodology and TouchTest form the blueprint for best-in-class continuous testing of mobile applications.

2. Other Mobile Testing Products Cannot Meet the Challenge

There are a few products on the market that provide mobile device testing and monitoring. These products are struggling to deliver results in the mobile testing space due to fundamental design problems in their approach. TouchTest was created in response to resounding feedback from the industry that the current solutions are failing at a time when they are needed most. This section addresses some of the more common issues that arise with other mobile testing products.

2.1 Problem – Jailbreaking of iOS devices is mandatory

Most mobile testing solutions require jailbreaking devices in order to test. This is a poor approach for an enterprise testing solution. Jailbreaking is the process of hacking a particular version of iOS to allow unauthorized use of apps and operating system modifications. Jailbreaking is not analogous to open source on an iOS device – it is the installation of a hacked and modified operating system kernel onto the device. This is a significantly different operating environment than what Apple shipped from the factory. Jailbreaking resulted in the first iOS worm, iKee, being unleashed on Apple devices. Apple’s original intent for sandboxing is more relevant than ever as they work to preserve their customer’s experience and security. Jailbreaking is also known for its association with application piracy.

Perhaps the most serious jailbreaking issue for software testers is the fact that every time a new version of iOS is released anyone using a solution built on public jailbreaking must wait weeks or months for a new public jailbreak to be released. This introduces a ‘dark period’ in which no testing can be done on a new version of iOS. Data shows that within the first week of a new iOS release 40% of Apple devices are upgraded to the new version. Within 30 days, more than 60% of all devices are upgraded. In the case of iOS version 5.1, as of the writing of this white paper more than 45 days after its release, there is still no public jailbreak available and automated testing has ground to a halt for companies whose testing products rely on jailbreaks.

Jailbreaking becomes more difficult with each release as Apple tries to prevent it from taking place. With the release of iOS 5.1, Apple introduced a Jailbreaking countermeasure called ASLR (Address Space Layout Randomization). This technology randomizes data on the RAM to prevent exploits. This is, in part, why no jailbreak has yet been released.

Jailbreaking is currently legal under exceptions to the Digital Millennium Copyright Act (DMCA). However this ruling is up for review in 2012, as it must be reviewed every three years. Apple is lobbying to make jailbreaking illegal and has stated that it will, at the least, invalidate the warranty.
This has been one of the only ways to automate iOS testing. Apple’s sandbox approach is in place to ensure application security and should be left intact. This approach is also inaccurate since, in order to be precise representation, it would require end user devices to be jailbroken as well. When using a jailbroken device, engineers are using a test bed that doesn’t accurately reflect the end user’s operating system and device configuration.

2.2 Problem – Tethering is required at all times

In the real world devices are on carrier and Wi-Fi networks, and applications are used while a device is not tethered to a computer. Most current testing solutions require a device to be connected to the computer at all times for recording and playback, creating a slow testing process with unreliable results. It is particularly difficult to allow for remote automation of a device on a carrier network, as wireless providers block incoming messages to a mobile device for security reasons.

2.3 Problem – Optical recognition is fragile for functional validation

Today’s commercial products for mobile testing make extensive use of optical recognition, typically by using a tether and hardware integration, and one solution even points a webcam at a device to take pictures. Validations are done with coordinates and optical recognition algorithms on screen regions. This is an extremely fragile approach to functional validation. The threshold for a pass/fail on this type of validation is around five pixels on the screen, and even worse when it comes to colors. This means that if a developer moves a button up or down, left or right, or even changes the color of a button, a test case will fail, causing the engineer to modify the test.

In the case of Android devices, consider how many makes and models there are with varying screen sizes. A 10.1” screen versus a 5” screen may mean writing different test cases for each device with validations for different areas of the screen. The same would apply for a universal (iPhone and iPad) app on iOS.

2.4 Problem – Not viable for continuous integration testing

Continuous integration is an ongoing and rapid process. This means that teams need to be able to react quickly when a build breaks or new tests need to be built. Other products struggle in all aspects of modern software development. As previously mentioned, they are usually thick-client-based, require code to be written or test cases to be designed in complicated interfaces, and tests cannot be invoked remotely with results reported in a standard format.

3. SOASTA’s Solution – TouchTest

TouchTest addresses all of these issues with a unique and innovative solution. The TouchTest library is compiled into a native application and a testable version is created. Among many advantages, by being inside the application TouchTest allows you to:

- Record gestures and interactions with 100% accuracy and at the same speed, velocity and precision that the mobile OS sees … on the real device.
- Perform validations based on the actual objects and application states present on the device.
- Change internal application states or operate on hidden objects that a user typically might not.

The following sub-sections detail how TouchTest addresses the specific challenges already discussed with other testing products.

3.1 No jailbreaking required for iOS devices

TouchTest requires no jailbreaking. iOS applications can be tested on devices in an Apple supported configuration. In addition, teams never have to wait for a public jailbreak to be released, which means no blackout periods after new versions of iOS are released.

3.2 Testing unleashed – no tethering required

Mobile users are “in the wild”, and many types of tests should be conducted in the wild as well. Carrier networks introduce varying levels of latency and connection behavior into app performance. In addition, record and playback should be possible from any location desired. TouchTest allows you to play back a test on a device from any location. TouchTest users can run tests on multiple devices from different continents at the same time, with the push of a button.
Note: one exception to this capability is continuous integration testing, which requires a device to be tethered for purposes of app deployment to the mobile device. This is due to Apple restrictions on deployment — see section 4 for more details.

3.3 Native level validations
The TouchTest driver is compiled into the application. By being at the native level, validations are done directly on objects themselves, not by using optical recognition of what is displayed on the screen. You can check to see if the text “Welcome” is actually present, where expected and spelled correctly. You can look for the button named “LoginButton” to see if it is visible on the screen (if that was the name given to it by the developer). If the button moves to the other side of the screen, or if the screen size is different, the test that checks to see if the button is present will still pass.

3.4 Full support for continuous integration
Just like all aspects of the CloudTest platform, TouchTest is built for speed; speed in test creation, execution and results analysis. Below are some examples of how TouchTest helps teams achieve speed in all areas of testing.

Record from real devices
TouchTest test cases are recorded by interacting with the mobile application just like a real user would … directly on the device under test.

Visual test creation and modification
As with all test creation in CloudTest, touch paths are created and modified in SOASTA’s patented visual test editor, known as the Clip Editor.

Web-based interface
Everything in CloudTest, including TouchTest, is done through a web browser. Engineers can access the TouchTest instance from anywhere that a web browser can reach it.

Analyze results in real time or post-test
CloudTest provides real-time analytics through its horizontally scalable, in-memory architecture. The same mechanism that SOASTA customers use to run tests in excess of one million virtual users allows TouchTest users to watch tests run in real-time, regardless of the number of devices included in the test. Step through a test on each device and monitor the status in real time, along with capturing pass/fail analytics for that test as well as performance metrics such as the time to complete gestures and transactions.
Integration with Jenkins via Plugin

TouchTest ships with a Jenkins plugin that allows for remote execution of any TouchTest composition. Results of those tests are delivered in standard JUnitXML format for viewing within Jenkins. If any error occurs, a link is provided in the Jenkins error details that will open the TouchTest result inside of a frame in Jenkins and let the engineer diagnose the failure.

The power of a unified platform for all testing

The CloudTest platform delivers 4 distinct types of testing – functional testing for both mobile and browser-based applications using real devices and browsers, and performance testing for both mobile and desktop web apps at the protocol simulation level. All test creation, execution and results analysis is done in the same browser-based CloudTest interface using a common visual test creation paradigm, real-time analytics, and the continuity of the same interface give an unparalleled level of power and flexibility to engineers.

Teams can run a single test composition that includes protocol-level tests with thousands of virtual users, dozens of real browsers, and real mobile devices with detailed analytics about each segment, both aggregated and split out at a detailed level.

4. The Blueprint for Mobile Application Automation

There are other challenges in mobile test automation that need to be solved for a mobile testing solution to be complete. One is getting a testable version of the application deployed to the mobile device with no manual intervention. Having to manually deploy the application in order to kick off a battery of automated tests defeats the purpose of continuous integration testing. TouchTest, working in concert with SOASTA’s methodology, establishes a blueprint for continuous integration testing of mobile apps. This section of this white paper details an end-to-end CI workflow with TouchTest and offers select best practices from the CloudTest Methodology.

With a typical web application, a Jenkins CI workflow that includes CloudTest would check out source code, build deploy to a test environment, and then execute the CloudTest compositions. Deploying web applications to test environments is available today as a feature of Jenkins. However, deploying apps to mobile devices is a problem that has not been solved industry-wide, and is a key feature of TouchTest. It allows testing to be done on deployed builds of a mobile app without manual involvement from the engineer.
4.1 TouchTest CI Architecture

When building iOS applications, your continuous integration slave for mobile testing is a Mac. Code is built and deployed to the mobile device through XCode. Using Jenkins as the example CI server, the workflow is as follows:

- Jenkins CI server wakes up the slave and instructs it to check out the source code.
- Code is built on the Mac slave using the XCode command line utilities.

These are the typical steps for automating the build of iOS applications. The following steps are unique to TouchTest:

- One or more iOS devices can be connected to the Mac slave via USB.
- The TouchTest deployment utility, as a step in the Jenkins job definition, will automate deployment of the app to the devices through XCode.
- Once deployed, devices will automatically connect to TouchTest and be ready to receive and execute tests.

After the application is deployed via TouchTest automation through Jenkins, execution and reporting can begin.

- Tests are executed.
- Results are fed into Jenkins for pass/fail reporting and post test analysis.

Results are fed back into Jenkins using the industry standard JUnitXML format for viewing. Any time a failure is encountered, the TouchTest Jenkins plugin allows the engineer to view the exact failure in TouchTest – within the scope of Jenkins without having to separately log in to CloudTest – search for the results and drill into the exact failure: a link takes you directly to the result and failure for quick analysis.
The CloudTest Methodology suggests two trains, one in each of the lab and production environments, that maintain a schedule of continuous testing in a lab with targeted production tests with go/no-go decisions made at each tollgate.

### 4.2 The CloudTest Methodology

The CloudTest Methodology, released in 2009, is the body of best practices for testing applications in a production environment, testing in a lab and in production with continuity between them, and testing on a routine schedule between both environments on two separate trains. Since developing the methodology, SOASTA has enabled numerous companies to achieve best-in-class results in their application testing operations.

The CloudTest Methodology focuses on execution, as evidenced in the diagram below. The largest pillar in the process, a focus on execution, means that more value is delivered from testing operations.

Within the execution phase, CloudTest and TouchTest enable rapid test cycles by accelerating test definition, design, execution and assessment. The visual development environment allows teams to create tests with unparalleled speed, as little to no code is written throughout this process. Execution can be done in an automated fashion via schedulers or continuous integration systems. As with all data in CloudTest, results are delivered in real time. This means that engineers can get immediate feedback by creating and manipulating dashboards in real time. They get instant visibility into application performance and stability.
Conclusion

Success in today’s fast-moving online world demands that companies have the right testing solution. Creating a poor customer experience can be catastrophic. Poor performance can result in lost revenue, failed customer retention, and a viral spread of poor brand reputation. The traditional players in the marketplace have been slow to respond with solutions that deliver a high level confidence that these problems will be avoided. That confidence comes from having a modern testing approach and the right tools for the job. With TouchTest, SOASTA delivers an approach and tool set that overcome the challenges associated with mobile application testing.

SOASTA is the leader in cloud testing and is at the forefront of modern application testing. SOASTA has received numerous awards and accolades, including recognition by Gartner as a Visionary Leader in the Magic Quadrant for Test Tools, and inclusion in Wall Street Journal’s list of Top 50 Venture Backed Company (out of 38,000 world-wide) two years in a row. SOASTA empowers companies to deliver the highest confidence possible in their online and mobile applications. Leading online names such as Chegg, Gilt Groupe, Hallmark, Intuit, Netflix, Zappos, and over 400 other SOASTA customers use CloudTest for their online and mobile applications … in addition to thousands of others in the CloudTest community using CloudTest Lite (free version) for their testing needs.

About SOASTA, Inc.

SOASTA’s mission is to ensure that today’s web and mobile applications and services perform in a high quality, scalable, and predictable manner. The company’s product, SOASTA CloudTest is available as an on-demand service in the Cloud or as an on-premise installation and it enables developers to test their web and mobile applications and services at an affordable price. The SOASTA CloudTest Platform supports Load, Performance, Functional and Web UI/Ajax testing for desktop web applications and native mobile applications. SOASTA is privately held and headquartered in Mountain View, California. For more information about SOASTA, please visit www.soasta.com.

Contact us:
866-344-8766
444 Castro St., Sixth Floor
Mountain View CA 94041
FAX: 650-210-4957

To learn more visit:
soasta.com
or email us at
info@soasta.com

©2012 SOASTA. All rights reserved. SOASTA, the SOASTA logo, and SOASTA CloudTest are trademarks of SOASTA. All other product or company names may be trademarks and/or registered trademarks of their respective owners.