Continuous Integration (CI) for Mobile Applications

Author:
Guy Arieli, CTO, Experitest

Table of Contents:

- What Continuous Integration Adds to the Mobile Development Process  2
- What is Continuous Integration?  3-4
- Why CI is Important for Mobile Applications  5
- How Mobile CI Differs from Standard Continuous Integration Solutions  6-7
- Putting it all Together: CI in Action  8
- Recommended Process for Mobile Continuous Integration  9-11
- The Tools You need for Optimal Continuous Integration  12

Abstract:
The mobile universe gets bigger at an ever increasing rate. At the same time, users demand newer versions faster, and with no errors. How does today’s business cover the rapidly expanding test area leaving room for no error in less time than before? Continuous Integration is the answer. Continuous Integration allows you to automate as much of the DevOps process as you can, reducing the time you need to deploy, or redeploy the latest version of your mobile application to market. The most challenging part of the process is mobile testing. An automated mobile application testing tool is a vital part of the CI process. Experitest's SeeTestAutomation integrates seamlessly into the main CI platforms. In this White Paper we will explore the CI process, how it improves the ROI of your DevOps, especially for mobile apps, and how you can leverage the entire spectrum of Experitest testing tools to make the most of your CI efforts.
What Continuous Integration Adds to the Mobile Development Process

Originally application development wasn't agile. Each developer would work on one piece of the puzzle, then they would put all the code together and throw a switch to see what would happen. The inevitable bug would happen and a team would have to spend painstaking hours locating the cause of the problem. Then you needed testing to see which solution worked, fixing, more testing, and finally the release. Applications were released once a year.

With the introduction of continuous integration (CI) everything can happen all at once. A CI platform monitors everything all at once, tracking every step in the process. When a new feature is needed, a developer codes the new feature in his own programming environment. Once he is done he updates the new feature to the CI platform, and passes it off to the quality assurance (QA) analyst. In some cases the tests will be run automatically. If there is something wrong, the tests results will immediately halt the development process to alert the DevOps team exactly where the application isn't working. If everything is working, the CI server can redeploy the new build to market.

The Waterfall process is a sequential design process. A development team will work on one stage in the process, and not move to the next until the current stage is completed. Once a step has been completed, developers won’t go back to a previous step. There’s little room for change or error, so the plan must be followed with precision.

To give the waterfall development process more flexibility, or to make it more agile, the Agile approach was developed. Instead of a sequential design process, the Agile methodology follows an incremental approach.

Developers start off with a design plan, and break it up into small projects. The work on the projects is done in weekly or monthly time periods. At the end of each project, tests are run. Bugs are discovered throughout the process and incorporated into the design as the application is being developed.

Continuous integration makes a DevOps team so agile they can go from one build a year to at least one build a day. This is vital because the platforms mobile applications execute on, such as devices, operating systems, networks, browsers, etc. are all continuously expanding their capacities. This demands that applications continue to expand their abilities to make the most of what each platform offers to continuously offer the user the best experience possible.
What is Continuous Integration?

Continuous Integration is an application development practice where members of a team integrate their work frequently. Each person integrates at least daily – leading to multiple integrations per day. A team of developers are all coding – and they have to update their code many times each day. They will update their code, or make new code for something, and upload it to a centralized Continuous Integration server where every individual step is integrated with the entire project.

The CI server has 2 roles:
1. Create a new build of your app based on each update.
2. Run the necessary tests for each build to confirm that the specific update has no errors.

This is the added value: Many developers can update their code all at once. The process evolves into an agile operations where instead of integration being one point where every developer puts their piece into the entire puzzle and then the whole thing is tested, each developer can constantly put their one piece into the build and the server will continuously test and update the new build to determine exactly what is working, and what needs to be fixed and tested further.

The advantage of the CI is that if one piece of code isn’t working, or one update is incompatible with a series of updates performed by other developers at the same time, the server will stop the build. If a test fails the CI process is stopped, and developers are told exactly at what point the new build failed.
What is Continuous Integration?

Bugs can be fixed immediately where they appeared, and developers can quickly adapt to a new coding reality based on incompatibilities between different code blocks being developed and updated. The testing phase accompanies the development phase, rather than waiting for it to be completed. If problems were only discovered after all the new builds were coded and put together as in the past, fixes would be harder to find and tougher to solve, requiring more time in the development process.

Continuous Integration enables continuous testing. As long as you are developing, testing, and verifying every step in a build, you can do the same for every update. You can complete a version of your application, then continue the process as you add updates to your app to keep up with constant updates to the platforms like operating systems, and devices that make these updates to your app necessary to compete in the marketplace. The continuous improvement of your app will maintain the highest standard of quality for your application before re-releasing it to market.
Why CI is Important for Mobile Applications

More than desktop applications, mobile demands speed. Where the standard desktop app could take from 4 to 6 months to develop, the mobile app completes the process in 4 weeks. End users expect faster response time. Once the application is developed, new updates need to be constantly added. Frequent external environment changes like new OS versions, and new devices demand apps constantly keep up. As a result, there is a high adoption of Agile methodology in R&D.

Mobile testing becomes the biggest challenge. Mobile testing is the longest part of the process because you need to test over at least 4 different mobile operating systems, hundreds of devices of varying shapes, sizes, and processing speeds, and you need to test these scenarios over different networks across the world. This demands Agility in your DevOps process.

As the test matrix for a standard mobile application development widens, the need, and return on investment (ROI) for continuous integration grows. Because the matrix of devices and OS and versions is so large, you must have CI. Not only do you need CI, you must also have test automation, and cloud testing.
How Mobile CI Differs from Standard Continuous Integration Solutions

It is not enough to have a process to build the application, you need a process to test it so you can maintain the app's high quality throughout the development process. What is the point of a new update if it will cause a function to stop working or it slows down the performance of the mobile app?

Mobile CI adds three dimensions which enhance your CI efforts: Smart device allocation, Dual application mode (instrumented or non-instrumented), and Parallel execution.

**Smart Device allocation.**

Once the mobile application is ready to be tested you need to allocate the mobile device you want to test it on. These devices are not sitting in one fixed location. They are shared with other testers. You have allocate each device between the manual testers and the CI server. The CI server will conduct tests on devices that are shared with users so you need to make sure devices are available when tests are automatically being run, and that the CI server is configured to run tests when the QA managers know to have the devices available to them. SeeTestCloud has a reservation system that will enable mobile testers to select the devices they need, check if and when they are available, and reserve the devices for the times they are going to test. A QA manager is given privileges to reassign devices to testers depending on the priority of the project.
How Mobile CI Differs from Standard Continuous Integration Solutions

**Dual Application Mode**

You have the ability to instrument the mobile application. This is where code will be injected in the app that gives you more control over the manipulation of the app. An instrumented mobile application gives you more specifics on the status of functions, objects, and methods of the applications as it is being tested. SeeTest's Object Spy will instrument a mobile application to enable you to identify the functions, objects, and methods of a mobile application so you can test it with even greater precision.

**Parallel execution**

With so many devices an app needs to be tested over, you cannot test one at a time. Tests must be conducted on many devices at the same time. An app needs to be tested over a device running Android, and a device running iOS, and two tablets of varying sizes all at once. The DevOps team needs to know how the app works on all these platforms and environments to guarantee that the app will work well anywhere, anytime, and on any device. You can have up to 100 devices all at once running parallel tests for the same mobile application. SeeTestAutomation has parallel execution capabilities. You can test on as many phones are connected to your computer all at once. Along with SeeTestCloud, you can also test over as many phones are in the mobile testing lab, performing mobile testing over tens, even hundreds of devices in parallel execution.
The developer enters the source code into the source code repository. The CI is triggered to work once a day, or as periodic as you want. It takes all the new code blocks, updates everything, and rebuilds your mobile application based on all the changes. The process is automated. It can happen in the middle of the night while the office is closed.

Once the CI builds the app, it will transfer the app to an executor agent for mobile testing.

The executor agent passes the app to all the devices it needs in order to conduct adequate mobile testing Using SeeTestAutomation, you can instruct the tool to install the application on all of the mobile devices at once. The tests will be run in parallel over a matrix of devices. They can be devices connected directly to the executor agent (the platform the untested version of your app is being run on), or they can be connected through the cloud.

Once the tests are run, you get the test results. They will go straight to the developer, who will see the results, determine exactly what went wrong, and make the necessary fixes. The results can go directly to the CI server. If the mobile application passed all the tests, the CI can move the mobile app to the app store for redeployment to market. SeeTest tools integrate into the main CI environments: JIRA, QC, and Jenkins. Reports can be generated from these servers with SeeTest fully integrated into them.
Recommended Process for Mobile Continuous Integration

First build the mobile app. Then prepare it for automation.

**Build the app**

The development team adds their part of the code to the CI environment. This process happens on a daily basis. Then the CI server is creating an automatic build of the app. The output of the process is a URL. The CI gives you a URL where you go to download the application by the executer.

**Prepare the Application**

There can be two methods to prepare the application – working with the application as is (non instrumented) the other option is running the Instrumented mode which let’s you a wider ability to control your application. Your test automation tool should enable unattended, on the fly instrumentation. It should perform instrumentation on the same machine that is being used for automation.

To instrument the app simply include code in the app that gives you more control over the manipulation of it. The entire process should be automated so if you are going to instrument, you need a testing tool that does this on the spot, and not manually.

For the Continuous Integration server in mobile you want to have the instrumentation done on the windows machine that runs the automation so you don’t need to switch machines to conduct instrumented testing. This is a must! Anything less adds a manual step to the process. SeeTest products include on the fly instrumentation which solves the problem.
Recommended Process for Mobile Continuous Integration

Reserve the Right Devices without Creating a Bottleneck.

Once you are conducting testing on a cloud of devices, it can be up to 100-200 different devices. However, these devices are also used for other projects. Other apps are using your cloud to do their own testing. Manual testers reserve these devices for their own QA. You need to use these devices without interrupting other projects, or having other projects interrupt your CI process.

To dynamically reserve a suitable device, you need a dynamic query so you can get the right device at the right time. One method is to set criteria which requests a set of devices that meet these requirements, rather than reserve a specific device.

For example, you can query 'android' and 'version number > 4.2' and 'manufacturer=’htc,’ rather than requesting a specific device. Setting criteria says to the cloud, "find me any device within this range of options," enabling you to proceed with automation even if the specific device you want isn't available.

This reduces the time it takes to get your new build onto the right device and underway in the testing process. You can create several queries to make sure you have sufficient test coverage without limiting the options or creating testing bottlenecks. The SeeTestCloud Smart Reservation system makes your queries dynamic, empowering you to request from a range of potential devices, rather than just one.

Deploy your application on the device you selected.

Now that you have a URL for the app, you need to download the new build (which nobody has ever used before) onto the device for testing. You need full control of the device and of the application installation.

You need control over the following on the device to make it happen:

Device reboot: you need to be able to clean the device. Application clear data: you don't want existing application data, specifically from the previous build, to still be on the device. It needs to be wiped clean for testing the new build. Working with the settings: you want to go out of the application and control the wifi configuration. You want to be able to move from the application to the browser and back. You need to control parts of the device not associated with your application. You need this for both Android and iOS. Uninstallation of applications: you need to be able to uninstall the previous build of your app, install the new build, and launch the new version of the app.

You need to confirm that all of these steps can be done automatically. These commands are all part of the SeeTest Suite of products.
Recommended Process for Mobile Continuous Integration

**Tests & Execution**

You need the capability to run the tests in parallel on different devices. They can be multiple devices directly connected to your machine by USB cable, or several devices connected by the cloud.

There needs to be a decoupling between the CI server and the execution machine, and the queue of tests that are waiting for devices need to be available. We need to do load balancing of test execution. We are likely running tests on 30-40 devices.

If you have enough devices then your test execution will go quickly. If you don't have enough devices, you will likely have to wait for the devices to become available, and the process will take longer.

The last and very important part is the ability to collect the results and analyze them. The monitoring phase lets you drill down into the cause of the issues you are running into.

You want to know what caused the tests to fail. You need screen shots of the execution. You want to know where exactly it went well, and where exactly it didn't. This requires screen shots of every step of the test. SeeTest's reporting will give you detailed screen shots of every step in the process, which let's you identify what worked, and what needs fixing.
The primary tool you will need to include mobile testing in your Continuous Integration process is an automated mobile testing tool. You will need to create test scripts, and run them automatically on parallel devices. Your automation tool will have to run on all major ALM environments and be able to integrate into the major CI platforms like Quality Center, and Jenkins.

You will also need a cloud testing tool that enables you to build a mobile device lab. A cloud tool that enables you to connect to all your devices, reserve the devices you need, and that enables a manager to prioritize devices according to different projects.

Your testing tool must have monitoring capabilities that include generating a screen shot of every step, determining what exactly worked, and what didn’t. The Continuous Integration process will halt if a new build wasn’t developed properly. It will indicate which build needs to be fixed. It is up to the testing tool to show you what step within that build failed, enabling you to pinpoint exactly where you need to improve your mobile application to complete the new build.

Experitest’s SeeTest Suite, which features SeeTestAutomation and SeeTestCloud were designed to include all of these features. With SeeTest, a DevOps process that originally took months can be reduced to hours in identifying a problem in a new build and moving it towards a rapid completion.