Continuous Integration (CI) is well-known for desktop applications, but could it also work for mobile applications? This whitepaper will answer this question and outline both the challenges and solutions to that problem, based on our experience.
What is Continuous Integration?

“Continuous Integration is a software development practice where members of a team frequently integrate their work. Usually, each person integrates at least once a day, leading to multiple integrations per day. Every integration is verified by an automated build (including test) to detect integration errors as quickly as possible.” —Martin Fowler

Martin Fowler is talking about the three essential phases within CI:

1. integration
2. building
3. testing

A team is integrating their work, which leads to a build that will be tested.

Integration

Integration is the starting point in the whole CI process chain. The development team is working on software that is globally available to all the team members. Every team member can access—and change—the source code and make those changes available to all the other members.

Team members use a source code management (SCM) tool, such as Subversion or Git, to accomplish this. An SCM tool contains all the source code, the information about how to build the software and the appropriate tests for the software. This enables each team member to check out the source code to his or her machine, build the software and run tests—all within a couple of minutes.

Building

The source code repository contains everything required to create or build the software, so it is simple enough to create the software on any machine. In the CI process, software building is centralized on a dedicated machine—the build server.

The build server is responsible for triggering a build whenever somebody has integrated code changes, archived the build artifacts, run the tests or provided the build quality information to the team, especially the team member responsible for the build.

As a mobile application is generally available on several platforms, such as iOS, Android, Windows Phone, BlackBerry and Symbian, the build process is different to that of a desktop application. To overcome the challenge of building an application for five platforms, it is best practice to only check out the source-code for a specific platform and build the application for that platform. This practice has several advantages:

1. Ensures that a check in for an Android project will not trigger an iOS project build.
2. The flexibility to assign the build to specific build nodes—i.e., tying an iOS-project to a Mac build node and an Windows Phone project to a Windows build node.
3. Reduced build duration, as applications are not built for all platforms.

The result of the build process is the application itself. The verification process, to determine if the application works as intended, is the responsibility of the third phase of the CI process—the testing.

Testing

Testing isn’t perfect, but it gives a good indication of the build quality. The key in continuous integration is that the testing is automated and can be instigated from a simple command, which implies that all the test-machines are configured the same.

---

Testing is a science. The challenge is to find the right balance of what has been tested, to what extent and how it has been tested. Subscribers to the Agile Testing Quadrants² focus on the test categories that can be automated. Manual tests cannot be part of the CI process and must be applied in a different way.

Nevertheless, developers with a reduced set of testing categories still face the challenge of working with the provided test set, a combination of unit and functional tests, that run as long as the suggested duration of 10 minutes³. The usual strategy is to implement a staged build.

A staged build is a succession of multiple builds executed in sequence. For example, in a two stage build, the first build is triggered right after the check-in and runs a reduced test set, which is a compromise between the execution time and the importance of executed tests.

This reduced test set should cover the application’s critical workflows. All the other tests, which take much longer, will be executed after the tests within the initial build have been successfully completed.

**Deployment**

An additional challenge is the deployment of the application to its testing device. While all the tests can be executed in an emulator rather than on real, physical devices, it is an industry requirement to do so. Only testing on the device itself will ensure that the tests reflect the real user-experience and consequently deliver value. Many companies don’t allow testing on modified devices, so it is forbidden to jailbreak or root a device to make it testable.

Clearly, it is not possible to test on all available devices. The strategy must be to pick the correct set of devices to be tested. If the question is ‘does the screen size change the behavior of the application?’ then a tablet and a phone must be in the test set. If other criteria include the capabilities of the hardware, add a low-end and a high-end device. As a rule of thumb, eight physical devices will cover 80% of all the use cases.

But how can testers deploy the mobile application to the device in an automated way? Manual would be easy, as several tools will facilitate this. Unfortunately, most have no command line interface, which would enable automation. The solution is to use a mobile testing tool with that capability.

Micro Focus® Silk Mobile™ transfers an apk-file to the mobile device and then executes the provided use case. After the test-execution, a detailed report of what is happening should be part of the testing-step. This reports back to the developer if everything works as expected or where problems have occurred.

With that the whole CI process is completed and can be applied for mobile applications in the same way as for desktop applications.

**How-to in Practice**

This chapter shows how the CI process works in practice. The content is based on a blog entry⁴.

Different tools have been used for each phase of the CI process:

1. **Integrate**: Git⁵ is the source code management tool. It contains all the source code and tests.
2. **Build**: The CI build server Jenkins⁶ in combination with Ant⁷, Maven⁸ and the mobile SDKs have been used to build the mobile application.
3. **Test**: Silk Mobile⁹ has been used for testing the mobile application.

**Building the Mobile Application**

Building a mobile application requires the SDKs for the specific platform. The SDK must be present on the developer machine and the build server. All the other code must have been checked into the source code repository.

As an example, I used an Android application, available at https://code.google.com/p/eribank/. The mobile applications will be created by simply executing the Ant-build file. The check-out and the build-process will then be automated in a first Jenkins-Job.

---

⁵ Git, http://git-scm.com/
⁶ Jenkins, http://jenkins-ci.org/
⁷ Ant, http://ant.apache.org/
⁸ Maven, http://maven.apache.org/
The result of the build process is a signed apk file that can be deployed to your device or submitted to the appstore.

**Testing the Application**

All the mobile test scripts are written in Java and use the Silk Mobile library to interact with the mobile device. The library itself also contains a method that deploys the application to the device. This is a major advantage—and perhaps should be a requirement—for the whole CI process. Otherwise testers have to interrupt the CI chain by manually installing the application.

To utilize several build nodes and achieve maximum test flexibility, I created a second Jenkins-Job to execute the Silk Mobile test-set. This job will be triggered automatically as soon as the initial (build)-job has been completed.

The tests are written in a very generic way, enabling testers to configure on which device the test should be executed via the command-line. The result of the test-set is a common junit-report that shows the failing, skipped or passed tests and historical data of the build quality.

This is the last step and concludes the CI process for mobile applications. Details of the practical use-case can also be found at: [http://community.microfocus.com/borland/test/silk_for_mobile/w/wiki/968.ci-for-mobile-applications.aspx](http://community.microfocus.com/borland/test/silk_for_mobile/w/wiki/968.ci-for-mobile-applications.aspx)