

Mobile

SDF certification

Mobile Application Delivery in a leading bank group in Europe

*Prepared for* **Internal purpose** **January 2018**

*Prepared by*

Sr. Associate

sjain91@sapient.com

**Revision and Signoff Sheet**

**Change Record**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Author** | **Version** | **Change reference** |
|  |  | 1.0 | Initial draft for review/discussion |
|  |  |  |  |

**Reviewers**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Version approved** | **Position** | **Date** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

In mobile application, for customer experience, we primarily focus on UI and usability. However, Data security, code maintainability, app crashes, memory and battery usage are also important factors while developing mobile application. Mobile application delivery is a tough nut to crack. Determining how to deliver mobile applications, all depends on your environment and your user’s needs. This document provides details of the mobile application delivery process used in one of leading banking group in Europe. This process has replaced the old process which was causing several delays and losses.

# Background

Initially The bank was having many issues in delivering any new feature to customers. One new feature used to take minimum 6 months in development and 1 year in deployment. This was affecting both the business and finances as you are investing lots of money in delivering one feature to customers.

Some of the issues in old process were:

* Environment issues (Server connection)
* Unnecessary delays in decisions
* Bad planning and execution
* Lack of documentation
* Manual testing
* Dependencies on various teams
* Duplication at code and files level
* Bad quality of code

The new process which evolved over three months to replace old process helped in developing features at faster rate and deploying them on time.

# Challenges

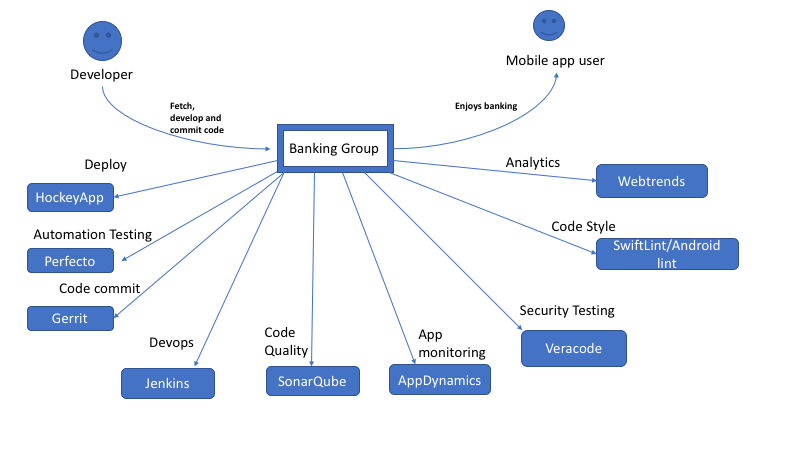
The real problem will become even clearer once you figure out who all your stakeholders are—that is, which functions or people might be affected by the new process. Here, we are talking about replacing old mobile application of a bank with completely brand new application. The old application was number one in finance market and was running successfully. Main Challenge was to convince business as this was going to affect their growth and also You need money to run any show. Here are the challenges :

* Stakeholder approvals – Includes business and finance
* Less clarity on business requirements
* Handling of different brands (4-5 different bank under umbrella)
* Collaboration between teams (Solution architects, Business analysts, UX and UI designers, tech leads, QA leads)
* Different approaches by different people
* Internal competitions
* Lack of developers and testers
* Security of customer data
* Old code analysis and knowledge

# Solution

Here, we are helping client to deliver high quality product on time. We are the delivery partners and we are committed to help them. The challenges were huge in number but we were able to solve them with the help of deep analysis and knowledge. Following are the objectives of new approach:

* Build a platform allowing us to improve quality & efficiency
* Engineer the best product for our customers’ needs
* Evolve our approach & technology to support the needs of the business
* Build an industry leading Engineering team
* Architect a future proof platform embracing a mobile-first outlook

****

# As shown in diagram, tools are used at different stages for designing, development, testing, analytics and deployment. Let’s talk about the approaches used in different stages:

1. **Designing and DOR review:** Each feature has been divided into epics and Business Analysts has to document all the stories in **JIRA**. **Confluence** was used to document each and every decision about the architecture of each component. Visual designers were the part of sprint teams and They provided inputs in backlog grooming. Here, **Zeplin** has been used extensively for designs.

* Fixed Font and color pallete has been used to decide for each brand. No change in them for any feature or journey. It helped in gaining uniformity across application. Like for Headline, there will be same color and font across all the journeys in application. It won’t change for any specific journey.
* Demo of custom components has been created to avoid confusion. This demo feature was added in application which can be seen only in non-prod version. It was helpful for developers to understand the functionality of components.

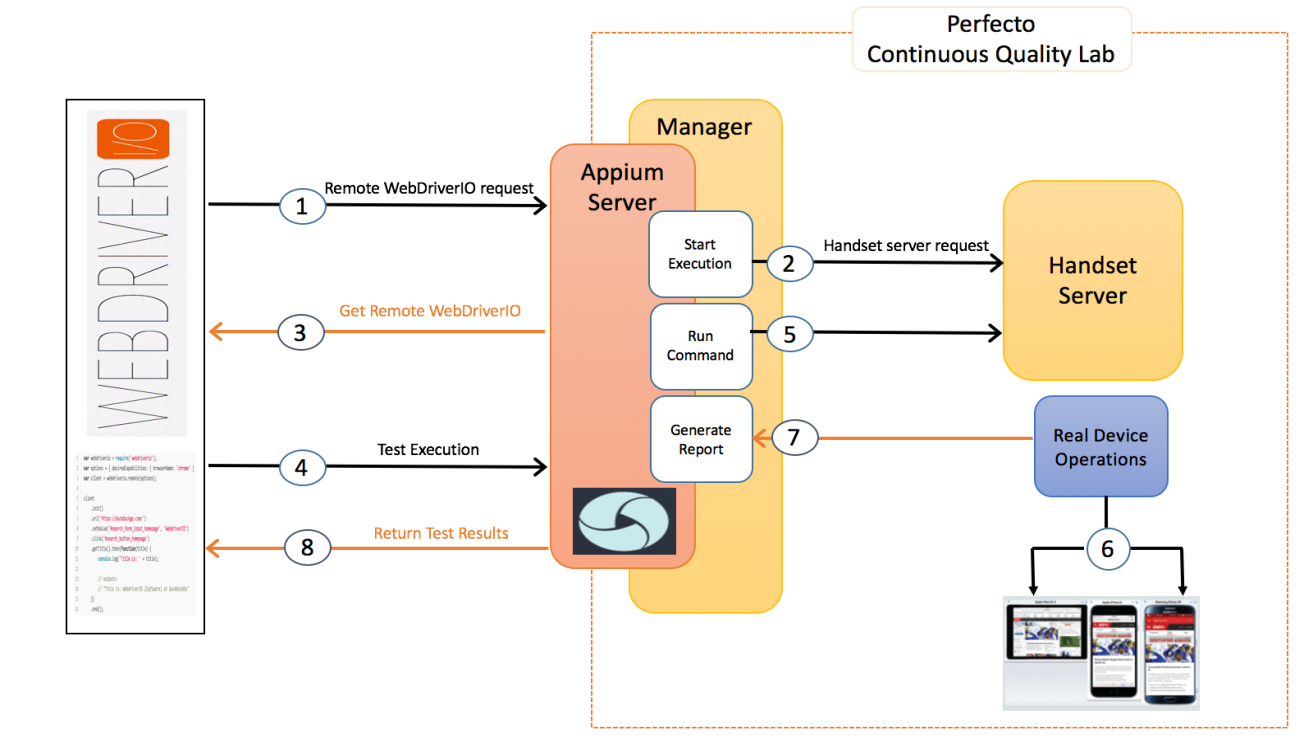
1. **Development:** Developers have to be very proactive in understanding story and designs. They need to understand the review process thoroughly as it goes through many levels. iOS developers have SwiftLint with xcode for compile time corrections. Whereas, android developers use android Lint to have structured quality of code**.**

* Usage of stub to avoid dependency on api. Generally, we have stub only to handle success scenarios. But here, we created stub for failure scenarios too. This removed the API dependency and also helped testers to understand negative scenarios.
* Stub has been stored in common repository for android and iOS. This reduces duplication in work as both needs json in the end. This repository was added as dependency in both the projects.
* There are many urls used in app for dev, QA, UAT and prod. These urls were also stored in common repository for iOS and Android. So, if one updates, other has no need to worry about the change. It should work automatically.
* We separated platform, business, UI as independent frameworks in application. These reduces dependencies as platform will not change for any brand. In future, if bank wants to make separate application, they can easily take out platform and use as base for development.
* Unit tests has been written for platform and business with very high code coverage. It’s the responsibility of developer to write unit test case if they are adding any code in platform and business frameworks. If they don’t add any test case, then Jenkins job shows code coverage which can highlight missing unit test cases.
* Once developer is done with particular story and raises the pull request. There is a job in Jenkins which assigns two reviewers to the pull request. A group of developer and reviewers will be created in Slack to discuss on review. This helps in faster reviews and code merge becomes easy.
* Sonarqube also runs as the part of Jenkins job for inspection of code quality and job will fail if the rules have not met. Developer has to take ownership of their code and get it passed through reviewers.

1. **Testing and DoD review:** Once the pull request is merged and builds are generated. QA team has to pick up the story for testing. Team is responsible for writing scripts using WebdriverIO framework. Scripts which would be enhanced on the basis of Stories, and would be executed as part of story testing.

* BDD scripts should be written and ready before the sprint starts. These scripts help developers to understand all the scenarios. Scripts will be executed using perfecto
* Optimized approach to test application against all the brands. This reduces manual testing efforts and saves testers time in executing same scenarios for all the brands. Copy deck automation and compatibility across multi devices (as huge numbers of devices are available for iOS and android in market)
* The Compatibility testing would be done via Automation script execution for the automated scripts on Perfecto Mobile Cloud against the attached Device list combination available on perfecto and rest which are not present in the perfecto cloud will be tested manually. Devices would be picked up at random level
* Webstorm is the IDE used for automation. WebDriverIO is a javascript based framework that sends requests to a Selenium Server through the WebDriver protocol and handles its response. It can be easily integrated with Cucumber-JS for writing scripts in BDD format
* Appium is an Automation Framework that provides capability to Test Native/Hybrid/Web apps against Real Devics, emulators and simulators. Perfecto is a Cloud based Test Automation solution that supports various Mobile and Web Platforms like iOS, Android, Safari and Firefox.
* Once the story is marked as done, feature branch containing scripts is merged in to master and regression is run during nightly CI builds. If there is a failure in regression, issue will be logged in Jira.

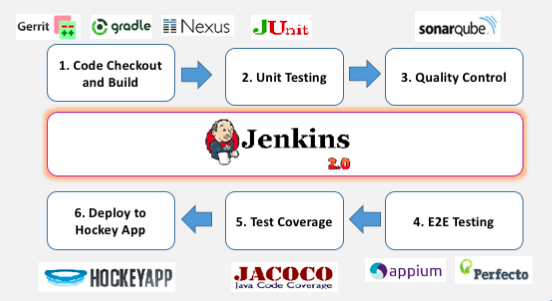
Automation testing approach is shown in below diagram.



1. **Analytics and Reporting:** Analyctis tool has been used to analyze the performance of app. Every page is being monitored to know the user journeys so that an improved version can be created on the basis of results. We are trying to use Tealium as well but currently we use Appdynamics and webtrends.

* Appdynamics monitors app usage and collects report using the data. This also helps in generating crash reports using logs.
* Webtrends monitors the web analytics for native and web journeys. This tool helps in deep analytics tracking and custom report building. This provides an optimized solution for generating reports.
* Veracode performs static analysis of code. This tool is helpful in code level security testing.

Here is the overall diagram for Devops pipeline used across all the stages:



# Summary

Usage of multiple tools and streamlined process helped in solving many problems discussed. Developers are more educated towards the code quality process. +2 reviewers are the chosen ones from different feature teams who made sure that only good quality of code goes in Master. QA team work towards automation of scenarios and provide certificate of quality. BA team is responsible for story grooming and provide answers to all the queries as much as possible. They made sure that every requirement is captured and documented. Each team effort, right usage of tools and structured process helped in creating impact on client and a better version of app has been launched.