If I were you App

Test Plan

Version 1.0

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 29/05/1015 | 1.0 | Initial Plan | Hendrik Boewer |
|  |  |  |  |

Test Plan

# Introduction

## Purpose

The purpose of the Iteration Test Plan is to gather all of the information necessary to plan and control the test effort for a given iteration. It describes the approach to testing the software, and is the top-level plan generated and used by managers to direct the test effort.

This *Test Plan* for the If I were you App supports the following objectives:

• Identifies the items that should be targeted by the tests.

• Identifies the motivation for and ideas behind the test areas to be covered.

• Outlines the testing approach that will be used.

• Identifies the required resources and provides an estimate of the test efforts.

## Scope

Unit Tests, Functional Testing and XX Testing are used.

## Intended Audience

The Test Plan is written for developers and customers of the If I were you App Project.

## Document Terminology and Acronyms

# Evaluation Mission and Test Motivation

The purpose of testing within the If I were you App is to secure a high quality of the software result and the corresponding code.

## Background

The background for testing in this case is the customers whish therefore and the self-created requirements to the result.

## Evaluation Mission

The misson for the evaluation effort includes all following concerns:

* find as many bugs as possible
* find important problems, assess perceived quality risks
* advise about perceived project risks
* certify to a standard
* advise about product quality, satisfy stakeholders
* advise about testing
* fulfill process mandates

# Target Test Items

The listing below identifies those test items⎯software, hardware, and supporting product elements ⎯that have been identified as targets for testing. This list represents what items will be tested.

* At least 30% of the developed code
* An user experience example for an initial installation
* XX

# Test Approach

[The Test Approach presents the recommended strategy for designing and implementing the required tests. Sections 3, Target Test Items, and 4, Outline of Planned Tests, identified **what** items will be tested and **what** types of tests would be performed. This section describes **how** the tests will be realized.

One aspect to consider for the test approach is the techniques to be used. This should include an outline of how each technique can be implemented, both from a manual and/or an automated perspective, and the criterion for knowing that the technique is useful and successful. For each technique, provide a description of the technique and define why it is an important part of the test approach by briefly outlining how it helps achieve the Evaluation Mission or addresses the Test Motivators.

Another aspect to discuss in this section is the Fault or Failure models that are applicable and ways to approach evaluating them.

As you define each aspect of the approach, you should update Section 10, Responsibilities, Staffing, and Training Needs, to document the test environment configuration and other resources that will be needed to implement each aspect.]

## Testing Techniques and Types

### Data and Database Integrity Testing

Not useful for this project, because the database is outsourced with the use of a backend as a service software.

### Function Testing

|  |  |
| --- | --- |
| Technique Objective: | Testing of Use Case flows |
| Technique: | Execute each use-case scenario’s individual use-case flows or functions and features, using valid and invalid data, to verify that:   * + the expected results occur when valid data is used   + the appropriate error or warning messages are displayed when invalid data is used |
| Required Tools: | The technique requires the following tools:   * All tools to create a build of the project (e.g. Android Studio, Gradle, Android SDK) * Cucumber Integration |
| Success Criteria: | A test coverage of the developed code of 30% in coherence with the unit testing |

### User Interface Testing

User Interface is realized by Unit test, because the originally purpose of testing logical functions with unit test is not possible within this project.

|  |  |
| --- | --- |
| Technique Objective: | [Exercise the following to observe and log standards conformance and target behavior:   * Navigation through the target-of-test reflecting business functions and requirements, including window-to-window, field-to- field, and use of access methods (tab keys, mouse movements, accelerator keys). * Window objects and characteristics can be exercised–such as menus, size, position, state, and focus.] |
| Technique: | [Create or modify tests for each window to verify proper navigation and object states for each application window and object.] |
| Oracles: | [Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination.] |
| Required Tools: | [The technique requires the Test Script Automation Tool.] |
| Success Criteria: | [The technique supports the testing of each major screen or window that will be used extensively by the end user.] |
| Special Considerations: | [Not all properties for custom and third-party objects can be accessed.] |

Einen der nachfolgenden Test entsprechend dem gewählten ausfüllen!

### Performance Profiling

[Performance profiling is a performance test in which response times, transaction rates, and other time-sensitive requirements are measured and evaluated. The goal of Performance Profiling is to verify performance requirements have been achieved. Performance profiling is implemented and executed to profile and tune a target-of-test's performance behaviors as a function of conditions such as workload or hardware configurations.

**Note**: Transactions in the following table refer to “logical business transactions”. These transactions are defined as specific use cases that an actor of the system is expected to perform using the target-of-test, such as add or modify a given contract.]

|  |  |
| --- | --- |
| Technique Objective: | [Exercise behaviors for designated functional transactions or business functions under the following conditions to observe and log target behavior and application performance data:  • normal anticipated workload  • anticipated worst-case workload] |
| Technique: | • [Use Test Procedures developed for Function or Business Cycle Testing.  • Modify data files to increase the number of transactions or the scripts to increase the number of iterations that occur in each transaction.  • Scripts should be run on one machine (best case to benchmark single user, single transaction) and should be repeated with multiple clients (virtual or actual, see Special Considerations below).] |
| Oracles: | [Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination.] |
| Required Tools: | [The technique requires the following tools:   * Test Script Automation Tool * an application performance profiling tool, such as Rational Quantify * installation-monitoring tools (registry, hard disk, CPU, memory, and so on * resource-constraining tools; for example, Canned Heat] |
| Success Criteria: | The technique supports testing:  • Single Transaction or single user: Successful emulation of the transaction scripts without any failures due to test implementation problems.]  • Multiple transactions or multiple users: Successful emulation of the workload without any failures due to test implementation problems.] |
| Special Considerations: | [Comprehensive performance testing includes having a background workload on the server.  There are several methods that can be used to perform this, including:  • “Drive transactions” directly to the server, usually in the form of Structured Query Language (SQL) calls.  • Create “virtual” user load to simulate many clients, usually several hundred. Remote Terminal Emulation tools are used to accomplish this load. This technique can also be used to load the network with “traffic”.  • Use multiple physical clients, each running test scripts, to place a load on the system.  Performance testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement.  The databases used for Performance Testing should be either actual size or scaled equally.] |

### Load Testing

[Load testing is a performance test that subjects the target-of-test to varying workloads to measure and evaluate the performance behaviors and abilities of the target-of-test to continue to function properly under these different workloads. The goal of load testing is to determine and ensure that the system functions properly beyond the expected maximum workload. Additionally, load testing evaluates the performance characteristics, such as response times, transaction rates, and other time-sensitive issues).]

[**Note**: Transactions in the following table refer to “logical business transactions”. These transactions are defined as specific functions that an end user of the system is expected to perform using the application, such as add or modify a given contract.]

|  |  |
| --- | --- |
| Technique Objective: | [Exercise designated transactions or business cases under varying workload conditions to observe and log target behavior and system performance data.] |
| Technique: | • [Use Transaction Test Scripts developed for Function or Business Cycle Testing as a basis, but remember to remove unnecessary interactions and delays.  • Modify data files to increase the number of transactions or the tests to increase the number of times each transaction occurs.  • Workloads should include (for example, Daily, Weekly, Monthly and so forth) Peak loads.  • Workloads should represent both Average as well as Peak loads.  • Workloads should represent both Instantaneous and Sustained Peaks.  • The Workloads should be executed under different Test Environment Configurations.] |
| Oracles: | [Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination.] |
| Required Tools: | [The technique requires the following tools:   * Test Script Automation Tool * Transaction Load Scheduling and control tool * installation-monitoring tools (registry, hard disk, CPU, memory, and so on) * resource-constraining tools (for example, Canned Heat) * Data-generation tools] |
| Success Criteria: | [The technique supports the testing of Workload Emulation, which is the successful emulation of the workload without any failures due to test implementation problems.] |
| Special Considerations: | • [Load testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement.  • The databases used for load testing should be either actual size or scaled equally.] |

### Stress Testing

[Stress testing is a type of performance test implemented and executed to understand how a system fails due to conditions at the boundary, or outside of, the expected tolerances. This typically involves low resources or competition for resources. Low resource conditions reveal how the target-of-test fails that is not apparent under normal conditions. Other defects might result from competition for shared resources, like database locks or network bandwidth, although some of these tests are usually addressed under functional and load testing.]

[**Note**: References to transactions in the following table refer to logical business transactions.]

|  |  |
| --- | --- |
| Technique Objective: | [Exercise the target-of-test functions under the following stress conditions to observe and log target behavior that identifies and documents the conditions under which the system **fails** to continue functioning properly  • little or no memory available on the server (RAM and persistent storage space)  • maximum actual or physically capable number of clients connected or simulated  • multiple users performing the same transactions against the same data or accounts  • “overload” transaction volume or mix (see Performance Profiling above)] |
| Technique: | • [Use tests developed for Performance Profiling or Load Testing.  • To test limited resources, tests should be run on a single machine, and RAM and persistent storage space on the server should be reduced or limited.  • For remaining stress tests, multiple clients should be used, either running the same tests or complementary tests to produce the worst-case transaction volume or mix. |
| Oracles: | [Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination.] |
| Required Tools: | [The technique requires the following tools:   * Test Script Automation Tool * Transaction Load Scheduling and control tool * installation-monitoring tools (registry, hard disk, CPU, memory, and so on) * resource-constraining tools (for example, Canned Heat) * Data-generation tools] |
| Success Criteria: | The technique supports the testing of Stress Emulation. The system can be emulated successfully in one or more conditions defined as stress conditions and an observation of the resulting system state during and after the condition has been emulated can be captured.] |
| Special Considerations: | • [Stressing the network may require network tools to load the network with messages or packets.  • The persistent storage used for the system should temporarily be reduced to restrict the available space for the database to grow.  • Synchronize the simultaneous clients accessing of the same records or data accounts.] |

### Volume Testing

[Volume testing subjects the target-of-test to large amounts of data to determine if limits are reached that cause the software to fail. Volume testing also identifies the continuous maximum load or volume the target-of-test can handle for a given period. For example, if the target-of-test is processing a set of database records to generate a report, a Volume Test would use a large test database, and would check that the software behaved normally and produced the correct report.]

|  |  |
| --- | --- |
| Technique Objective: | [Exercise the target-of-test under the following high volume scenarios to observe and log target behavior:  • Maximum (actual or physically-capable) number of clients connected, or simulated, all performing the same, worst case (performance) business function for an extended period.  • Maximum database size has been reached (actual or scaled) and multiple queries or report transactions are executed simultaneously.] |
| Technique: | • [Use tests developed for Performance Profiling or Load Testing.  • Multiple clients should be used, either running the same tests or complementary tests to produce the worst-case transaction volume or mix (see Stress Testing) for an extended period.  • Maximum database size is created (actual, scaled, or filled with representative data) and multiple clients are used to run queries and report transactions simultaneously for extended periods.] |
| Oracles: | [Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination.] |
| Required Tools: | [The technique requires the following tools:   * Test Script Automation Tool * Transaction Load Scheduling and control tool * installation-monitoring tools (registry, hard disk, CPU, memory, and so on) * resource-constraining tools (for example, Canned Heat) * Data-generation tools] |
| Success Criteria: | [The technique supports the testing of Volume Emulation. Large quantities of users, data, transactions, or other aspects of the system use under volume can be successfully emulated and an observation of the system state changes over the duration of the volume test can be captured.] |
| Special Considerations: | [What period of time would be considered an acceptable time for high volume conditions, as noted above?] |

### Security and Access Control Testing

[Security and Access Control Testing focuses on two key areas of security:

• Application-level security, including access to the Data or Business Functions

• System-level Security, including logging into or remotely accessing to the system.

Based on the security you want, application-level security ensures that actors are restricted to specific functions or use cases, or they are limited in the data that is available to them. For example, everyone may be permitted to enter data and create new accounts, but only managers can delete them. If there is security at the data level, testing ensures that “user type one” can see all customer information, including financial data, however, “user two” only sees the demographic data for the same client.

System-level security ensures that only those users granted access to the system are capable of accessing the applications and only through the appropriate gateways.]

|  |  |
| --- | --- |
| Technique Objective: | [Exercise the target-of-test under the following conditions to observe and log target behavior:   * Application-level Security: an actor can access only those functions or data for which their user type is provided permissions. * System-level Security: only those actors with access to the system and applications are permitted to access them. |
| Technique: | * [Application-level Security: Identify and list each user type and the functions or data each type has permissions for.]   + Create tests for each user type and verify each permission by creating transactions specific to each user type.   + Modify user type and re-run tests for same users. In each case, verify those additional functions or data are correctly available or denied. * System-level Access: [See Special Considerations below] |
| Oracles: | [Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination.] |
| Required Tools: | [The technique requires the following tools:   * Test Script Automation Tool * “Hacker” security breach and probing tools * OS Security Admin Tools] |
| Success Criteria: | [The technique supports the testing of for each known actor type the appropriate functions or data affected by security settings can be tested.] |
| Special Considerations: | [Access to the system must be reviewed or discussed with the appropriate network or systems administrator. This testing may not be required as it may be a function of network or systems administration.] |

### Failover and Recovery Testing

[Failover and recovery testing ensures that the target-of-test can successfully failover and recover from a variety of hardware, software or network malfunctions with undue loss of data or data integrity.

For those systems that must be kept running failover testing ensures that, when a failover condition occurs, the alternate or backup systems properly “take over” for the failed system without any loss of data or transactions.

Recovery testing is an antagonistic test process in which the application or system is exposed to extreme conditions, or simulated conditions, to cause a failure, such as device Input/Output (I/O) failures, or invalid database pointers and keys. Recovery processes are invoked, and the application or system is monitored and inspected to verify proper application, or system, and data recovery has been achieved.]

|  |  |
| --- | --- |
| Technique Objective: | [Simulate the failure conditions and exercise the recovery processes (manual and automated) to restore the database, applications, and system to a desired, known, state. The following types of conditions are included in the testing to observe and log target behavior after recovery:  • power interruption to the client  • power interruption to the server  • communication interruption via network servers  • interruption, communication, or power loss to DASD (Dynamic Access Storage Devices) and DASD controllers  • incomplete cycles (data filter processes interrupted, data synchronization processes interrupted)  • invalid database pointers or keys  • invalid or corrupted data elements in database] |
| Technique: | [The tests already created for Function and Business Cycle testing can be used as a basis for creating a series of transactions to support failover and recovery testing, primarily to define the tests to be run to test that recovery was successful.  • Power interruption to the client: power the PC down.  • Power interruption to the server: simulate or initiate power down procedures for the server.  • Interruption via network servers: simulate or initiate communication loss with the network (physically disconnect communication wires or power down network servers or routers).  • Interruption, communication, or power loss to DASD and DASD controllers: simulate or physically eliminate communication with one or more DASDs or controllers.  Once the above conditions or simulated conditions are achieved, additional transactions should be executed and, upon reaching this second test point state, recovery procedures should be invoked.  Testing for incomplete cycles uses the same technique as described above except that the database processes themselves should be aborted or prematurely terminated.  Testing for the following conditions requires that a known database state be achieved.  Several database fields, pointers, and keys should be corrupted manually and directly within the database (via database tools). Additional transactions should be executed using the tests from Application Function and Business Cycle Testing and full cycles executed.] |
| Oracles: | [Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination.] |
| Required Tools: | [The technique requires the following tools:   * base configuration imager and restorer * installation monitoring tools (registry, hard disk, CPU, memory, and so on) * backup and recovery tools] |
| Success Criteria: | The technique supports the testing of:   * One or more simulated disasters involving one or more combinations of the application, database, and system. * One or more simulated recoveries involving one or more combinations of the application, database, and system to a known desired state.] |
| Special Considerations: | * [Recovery testing is highly intrusive. Procedures to disconnect cabling (simulating power or communication loss) may not be desirable or feasible. Alternative methods, such as diagnostic software tools may be required. * Resources from the Systems (or Computer Operations), Database, and Networking groups are required. * These tests should be run after hours or on an isolated machine.] |

### Configuration Testing

[Configuration testing verifies the operation of the target-of-test on different software and hardware configurations. In most production environments, the particular hardware specifications for the client workstations, network connections, and database servers vary. Client workstations may have different software loaded⎯for example, applications, drivers, and so on⎯and, at any one time, many different combinations may be active using different resources.]

|  |  |
| --- | --- |
| Technique Objective: | [Exercise the target-of-test on the required hardware and software configurations to observe and log target behavior under different configurations and identify changes in configuration state.] |
| Technique: | * [Use Function Test scripts. * Open and close various non-target-of-test related software, such as Microsoft Excel and Word applications, either as part of the test or prior to the start of the test. * Execute selected transactions to simulate actors interacting with the target-of-test and the non-target-of-test software. * Repeat the above process, minimizing the available conventional memory on the client workstation.] |
| Oracles: | [Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination.] |
| Required Tools: | [The technique requires the following tools:   * base configuration imager and restore * installation monitoring tools (registry, hard disk, CPU, memory, and so on)] |
| Success Criteria: | [The technique supports the testing of one or more combinations of the target test items running in expected, supported deployment environments.] |
| Special Considerations: | * [What non-target-of-test software is needed, is available, and what is accessible on the desktop? * What applications are typically used? * What data are the applications running; for example, a large spreadsheet opened in Excel or a 100-page document in Word? * The entire system’s netware, network servers, databases, and so on, also needs to be documented as part of this test.] |

### Installation Testing

The Installation test is performed by an end user under observation of the project team.

|  |  |
| --- | --- |
| Technique Objective: | Exercise the installation of the target-of-test onto a required hardware configuration under the following conditions to observe and log installation difficulty:   * new installation: a used Android smartphone, never installed previously with If I were you App |
| Technique: | Installation performed by an End user without any knowledge about the project. |
| Required Tools: | The technique requires the following tools:   * An end user without previous knowledge * An Android Smartphone with Android 19 or higher * The project developers to observe the installation process * The executable installation file |
| Success Criteria: | The technique supports the testing of the installation of the developed product in one installation configurations. |

# Entry and Exit Criteria

## Test Plan

### Test Plan Entry Criteria

The execution of the Test Plan should begin directly after the first version of the Test Plan exists.

### Test Plan Exit Criteria

The execution of the Test Plan is complete as soon as the installation test ends successful and a test coverage of 30% of all developed code is reached. Whereby all of these test pass successful, too.

### Suspension and Resumption Criteria

No criteria exist, that the Test Plan can be suspended.

# Deliverables

## Test Evaluation Summaries

Test evaluation summaries for functional and unit testing are directly represented to the developer, who runs the tests/ starts a project and report on the.

An overall summary is presented on the project’s blog to give an overview about testing for every stakeholder.

## Reporting on Test Coverage

XX

## Incident Logs and Change Requests

Logs about test incidents are automatically produced to provide a possibility for the developer to get more information about incidents. Change Requests because of test incidents are seen as usual change request and are done in the project’s standard change request workflow with the corresponding forms.

(cf. <https://ifiwereyouapp.wordpress.com/2014/11/30/change-request/> )

# Testing Workflow

The testing workflow can be divided into two different test cases.

## Installation and XX Tests

These tests are executed only once for the case that the tests pass successful. Otherwise the tests will be repeated until the result becomes positive.

The installation and the XX test is scheduled immediately before the final release of the software product, so that only the final version has to be tested.

## Software Test

All other tests are executed within every build process. This gives a possibility to the developer to notice possible bugs or errors early and prevents the creation of a defective software.

# Environmental Needs

## Base System Hardware

The following table sets forth the system resources for the test effort presented in this *Test Plan*.

| **System Resources** | | |
| --- | --- | --- |
| **Resource** | **Quantity** | **Name and Type** |
| Database Server | Not needed, Backend as a service |  |
| Client Test Smartphone | 1 |  |
| —Include special configuration requirements |  | Any Android Smartphone with Android 19 or higher |
| Test Development PCs | 1 | Any PC with Windows as Operating system |

## Base Software Elements in the Test Environment

The following base software elements are required in the test environment for this *Test Plan*.

| **Software Element Name** | **Version** | **Type and Other Notes** |
| --- | --- | --- |
| Android SDK | 19 or higher |  |
| Android Studio |  |  |
| Facebook SDK |  |  |
| Gradle |  |  |

## Productivity and Support Tools

The following tools will be employed to support the test process for this *Test Plan*.

| **Tool Category or Type** | **Tool Brand Name** | **Vendor or In-house** | **Version** |
| --- | --- | --- | --- |
| Test Coverage Monitor or Profiler |  |  |  |

# Responsibilities, Staffing, and Training Needs

## People and Roles

This table shows the staffing assumptions for the test effort.

| **Human Resources** | | |
| --- | --- | --- |
| **Role** | **Minimum Resources Recommended**  **(number of full-time roles allocated)** | **Specific Responsibilities or Comments** |
| Test Manager  (Hendrik Boewer) | 1 | Provides management oversight.  Responsibilities include:   * planning and logistics * agree mission * identify motivators * acquire appropriate resources * present management reporting * advocate the interests of test * evaluate effectiveness of test effort |
| Test Designer  (Hendrik Boewer, Axel Mueller) | 2 | Defines the technical approach to the implementation of the test effort.  Responsibilities include:   * define test approach * define test automation architecture * verify test techniques * define testability elements * structure test implementation |
| Tester  (Axel Mueller, Hendrik Boewer, poss. Simon Tenbeitel) | 2 | Implements and executes the tests.  Responsibilities include:   * implement tests and test suites * execute test suites * log results * analyse and recover from test failures * document incidents |
| Implementer  (Hendrik Boewer, poss. Axel Mueller) | 1 | Implements and unit tests the test classes and test packages.  Responsibilities include:   * creates the test components required to support testability requirements as defined by the designer |

# Iteration Milestones

| **Milestone** | **Planned Start Date** | **Actual Start Date** | **Planned End Date** | **Actual End Date** |
| --- | --- | --- | --- | --- |
| Test Plan agreed | 20.05.2015 |  |  |  |
| Test coverage of 30% | 30.05.2015 |  |  |  |
| Software tests passed successful | 08.06.2015 |  |  |  |
| Installation test passed successful | 08.06.2015 |  |  |  |
| XX Test passed successful | 08.06.2015 |  |  |  |

# Risks, Dependencies, Assumptions, and Constraints

| **Risk** | **Mitigation Strategy** | **Contingency (Risk is realized)** |
| --- | --- | --- |
| Code coverage tool for Android does not exist | Extended and early search for possible code Coverage tool. Seek advice from other project teams, who create software for Android, too. | * Usage of other values and key figures to measure test effort. * Consider to skip measurement |

# Management Process and Procedures

The Test Plan can be extended and edited by every developer of the project with an appropriate version renewal.

## Measuring and Assessing the Extent of Testing

The extent of testing is measured in form of code coverage.