Politecnico di Milano
Computer Science and Engineering

Project of Software Engineering 2

# ***Integration Test Plan Document***

*Authors:*
Antonio Iannacci - 854157
Daniele Romanini - 854732
Federico Seri - 854032

*Reference Professor:* Mirandola Raffaela

# TABLE OF CONTENT

# 1. Introduction

## 1.1. Revision History

19-01-2016: Version 1.0

## 1.2. Purpose and Scope

The purpose of the integration test plan is to describe the necessary tests to verify that all of the components of *myTaxiService* are properly assembled. Integration testing ensures that the unit-tested modules interact correctly.
The team that will perform integration test should read this document.

## 1.3. List of definitions and abbreviations

- Driver: A software component or test tool that replaces a component that takes care of the control and/or the calling of a component or system.
- CI: Component Integration
- SI: System Integration

## 1.4. List of reference documents

- **Project description:**
  Assignment 1 and 2 (Section 2: The problem – MyTaxiService)
  https://goo.gl/pr652J

- **RASD:**
  RASD – MyTaxiService – Iannacci_Romanini_Seri.pdf
  *https://github.com/daler3/se2project/blob/master/Deliveries/RASD - MyTaxiService - Iannacci_Romanini_Seri.pdf*

- **Design Document:**
  Design Document – MyTaxiService – Iannacci_Romanini_Seri.pdf
  *https://github.com/daler3/se2project/blob/master/Deliveries/Design Document - MyTaxiService - Iannacci_Romanini_Seri.pdf*

- **Documentation of tools planned to be used for testing:**
  - Mockito: http://mockito.org/
  - Arquillan: http://arquillian.org/
  - JMeter: http://jmeter.apache.org/
  - JUnit: http://junit.org/

# 2. Integration Strategy

## 2.1. Entry Conditions

- Database drivers must be on the Server machine
- Database must have all the needed tables
- Functions must have been unit tested
- The Server and the client must be connected to a network

## 2.2. Elements to be integrated

Referring to the Design Document (section 2.3), we identified the following subsystems:

- Call: It is composed by the classes: Call, User and TimeDeamon.
- SharedCall: It extends the functionality of Call and it is composed by the entity SharedCall, SharedSet, User_TSharing and Call Recognizer.
- Zone: It is composed by the components: Zone, TaxiDriver and QueueManager.
- Server: It is composed by the component: server class and database.

## 2.3. Integration Testing Strategy

We choose to apply bottom-up strategy for testing: after each component at lower hierarchy has been tested, we proceed to test other components that rely upon these.

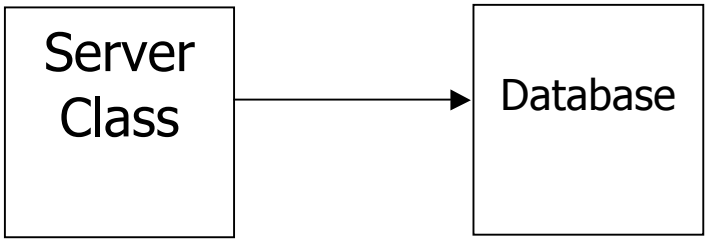
After having built the subsystems named in section 2.2, we integrate them, making interacting each other. The relations among the subsystems can be found at section 2.3 of Design Document. (In order to see the specific functions/methods called in the classes, section "2.7 – Component Interfaces" of Design Document can be consulted).

## 2.4. Sequence of Component Integration

### 2.4.1. Software Integration Sequence

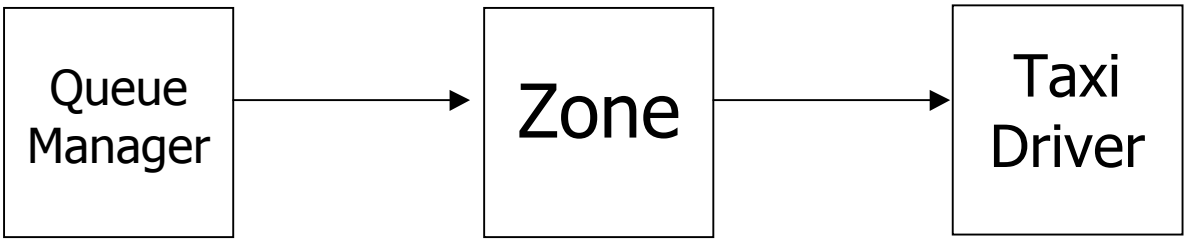
**Integration test of "Server" subsystem**

| ID | Integration Test | Paragraphs |
|---|---|---|
| CI1 | Server Class -> Database | 3.1.1 |



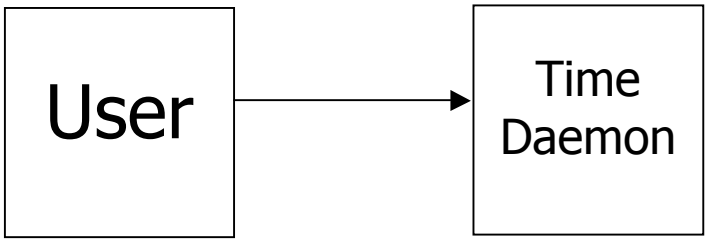

**Integration test of "Zones" subsystem**

| ID | Integration Test | Paragraphs |
|---|---|---|
| CI2 | Queue Manager -> Zone | 3.1.2 |
| CI3 | Zone -> Taxi Driver | 3.1.3 |



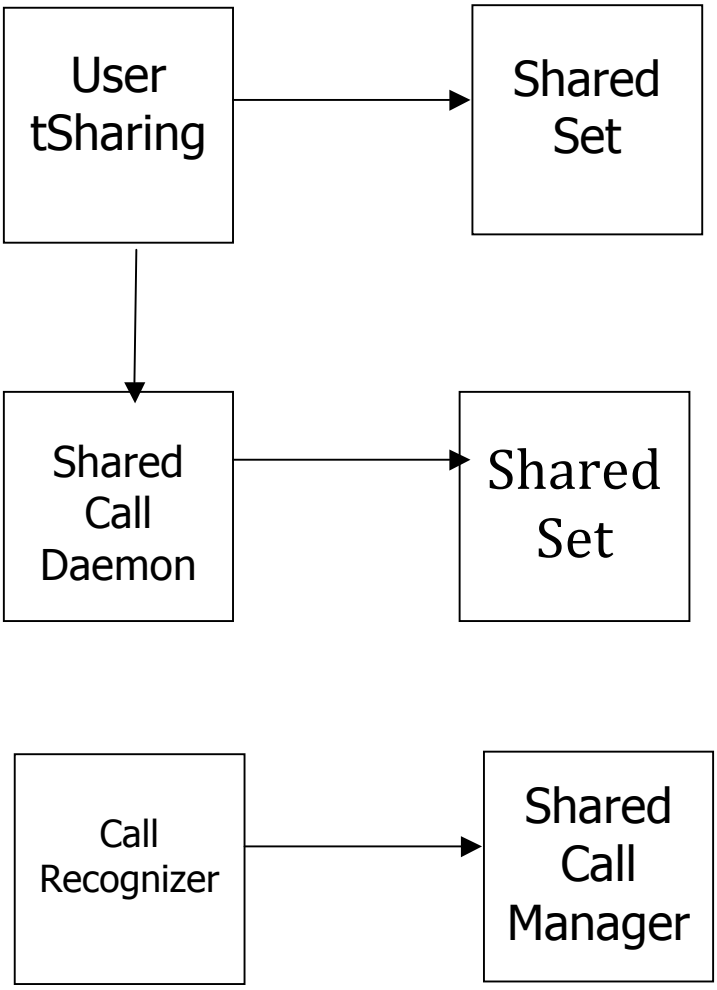

**Integration test of "Calls" subsystem**

| ID | Integration Test | Paragraphs |
|---|---|---|
| CI4 | User -> Time Daemon | 3.1.4 |



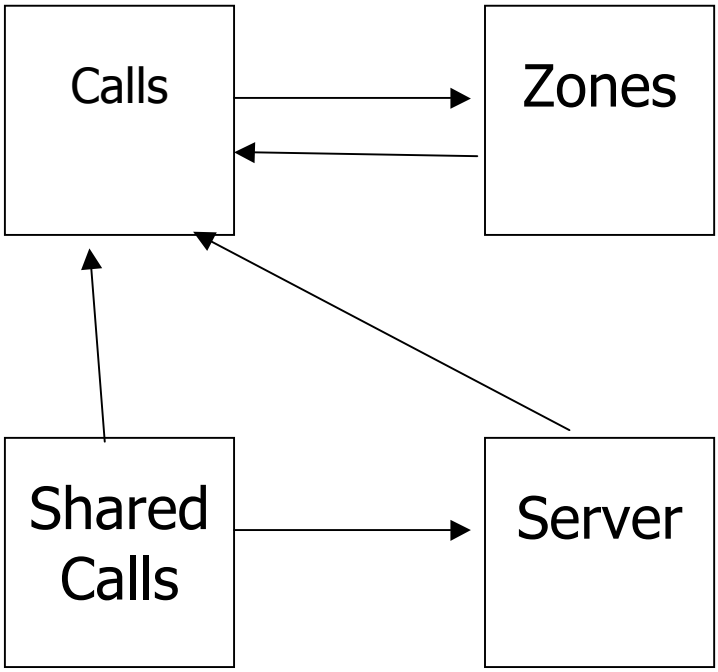
**Integration test of "Shared Calls" subsystem**

| ID | Integration Test | Paragraphs |
|---|---|---|
| CI5 | User_tSharing -> SharedSet | 3.1.5 |
| CI6 | User_tSharing -> SharedCall Daemon | 3.1.6 |
| CI7 | SharedCall Daemon -> SharedSet | 3.1.7 |
| CI8 | Call Recognizer -> SharedCall Manager | 3.1.8 |



### 2.4.2. Subsystem Integration Sequence

| ID | Integration Test | Paragraphs |
|---|---|---|
| SI1 | Calls -> Zones | 3.3.1 |
| SI2 | Zones -> Calls | 3.3.2 |
| SI3 | Shared Calls -> Calls | 3.3.3 |
| SI4 | Shared Calls -> Server | 3.3.4 |
| SI5 | Server -> Calls | 3.3.5 |



# 3. Individual Steps and Test Description

## 3.1. Component Integration

### 3.1.1. CI1

| **Test Case Identifier** | **CI1T1** |
|---|---|
| **Test Item(s)** | Server Class -> Database |
| **Input Specification** | Create typical Server input |
| **Output Specification** | Check if Database has been modified properly |
| **Environmental Need** | Server Driver; Database Driver; Network Connection available; |

### 3.1.2. CI2

| **Test Case Identifier** | **CI2T1** |
|---|---|
| **Test Item(s)** | Queue Manager -> Zone |
| **Input Specification** | Create typical Queue Manager input |
| **Output Specification** | Check if the correct methods are called in Zone |
| **Environmental Need** | Queue Manager Driver |

### 3.1.3. CI3

| **Test Case Identifier** | **CI3T1** |
|---|---|
| **Test Item(s)** | Zone -> Taxi Driver |
| **Input Specification** | Create typical Zone input |
| **Output Specification** | Check if the correct methods are called in Taxi Driver |
| **Environmental Need** | CI2 succeeded |

### 3.1.4. CI4

| **Test Case Identifier** | **CI4T1** |
|---|---|
| **Test Item(s)** | User -> Time Daemon |
| **Input Specification** | Create typical User input |
| **Output Specification** | Check if the correct methods are called in Time Daemon |
| **Environmental Need** | User drivers |

### 3.1.5. CI5

| **Test Case Identifier** | **CI5T1** |
|---|---|
| **Test Item(s)** | User_tSharing -> SharedSet |
| **Input Specification** | Create typical User_tSharing input |
| **Output Specification** | Check if the correct methods are called in Shared Set |
| **Environmental Need** | User_tSharing drivers |

### 3.1.6. CI6

| **Test Case Identifier** | **CI6T1** |
|---|---|
| **Test Item(s)** | User_tSharing -> SharedCall Daemon |
| **Input Specification** | Create typical User_tSharing input |
| **Output Specification** | Check if the correct methods are called in Shared Call Daemon |
| **Environmental Need** | User_tSharing drivers |

### 3.1.7. CI7

| **Test Case Identifier** | **CI7T1** |
|---|---|
| **Test Item(s)** | SharedCall Daemon -> SharedSet |
| **Input Specification** | Create typical SharedCall Daemon input |
| **Output Specification** | Check if the correct methods are called in Shared Set |
| **Environmental Need** | CI6 succeeded |

### 3.1.8. CI8

| **Test Case Identifier** | **CI8T1** |
|---|---|
| **Test Item(s)** | Call Recognizer -> SharedCall Manager |
| **Input Specification** | Create typical Call Recognizer Daemon input |
| **Output Specification** | Check if the correct methods are called in SharedCall Manager |
| **Environmental Need** | Call Recognizer drivers |

## 3.2. Component Integration – Test Procedures

### 3.2.1. TPC1

| | |
|---|---|
| **Test Procedure Identifier** | TPC1 |
| **Purpose** | • |
| **Procedure Steps** | Execute: CI1 |

### 3.2.2. TPC2

| | |
|---|---|
| **Test Procedure Identifier** | TPC2 |
| **Purpose** | This test procedures verifies whether the QueueManager:<br>• Can find the zone corresponding to a call<br>• Can access the taxi-queue of the zone corresponding to a call<br>• Can find the first available taxi-driver in the call zone<br>• Can handle the assignment of a taxi driver to a call<br>• Can handle the taxi-driver response. |
| **Procedure Steps** | Execute CI2 before CI3 |

### 3.2.3. TPC3

| | |
|---|---|
| **Test Procedure Identifier** | TPC3 |
| **Purpose** | This test procedures verifies whether the User:<br>• Can book a call |
| **Procedure Steps** | Execute: CI4 |

### 3.2.4. TPC4

| | |
|---|---|
| **Test Procedure Identifier** | TPC4 |
| **Purpose** | This test procedure verifies whether the classes related to SharedCall extensions work properly.<br>In particular we test:<br>• If a generic call is recognized as shared or not;<br>• If a User making a Shared-Call is assigned properly to a Taxi;<br>• If booked Shared Call are managed properly;<br>• If the appropriate fare is calculated for each user that has made a Shared Call |
| **Procedure Steps** | Execute: CI5 and CI6; then CI7; finally CI8 |

## 3.3. Subsystems Integration

### 3.3.1. SI1

| | |
|---|---|
| **Test Case Identifier** | **SI1T1** |
| **Test Item(s)** | Calls -> Zones |
| **Input Specification** | Create typical Calls input |
| **Output Specification** | Check if the correct methods are called in Zones |
| **Environmental Need** | User driver |

### 3.3.2. SI2

| | |
|---|---|
| **Test Case Identifier** | **SI2T1** |
| **Test Item(s)** | Zones -> Calls |
| **Input Specification** | Create typical Zones input |
| **Output Specification** | Check if the correct methods are called in Calls |
| **Environmental Need** | Queue Manager driver |

### 3.3.3. SI3

| **Test Case Identifier** | **SI3T1** |
|---|---|
| **Test Item(s)** | Shared Calls -> Calls |
| **Input Specification** | Create typical Shared Calls input |
| **Output Specification** | Check if the correct methods are called in Calls |
| **Environmental Need** | Shared Set drivers |

### 3.3.4. SI4

| **Test Case Identifier** | **SI4T1** |
|---|---|
| **Test Item(s)** | Shared Calls -> Server |
| **Input Specification** | Create typical Shared Calls input |
| **Output Specification** | Check if the correct methods are called in Server |
| **Environmental Need** | Call Recognizer Drivers |

### 3.3.5. SI5

| **Test Case Identifier** | **SI5T1** |
|---|---|
| **Test Item(s)** | Server -> Calls |
| **Input Specification** | Create typical Server input |
| **Output Specification** | Check if the correct methods are called in Calls |
| **Environmental Need** | SI4T1 succeeded |

## 3.4. Subsystem Integration – Test procedures

### 3.4.1. TPS1

| | |
|---|---|
| **Test Procedure Identifier** | TPS1 |
| **Purpose** | This test procedure verifies whether the subsystems "Calls" and "Zones" can interact each other. In particular we test:<br>• If the whole call procedure made by a user is properly managed; |
| **Procedure Steps** | Execute SI1 and SI2 |

### 3.4.2. TPS2

| | |
|---|---|
| **Test Procedure Identifier** | TPS2 |
| **Purpose** | This test procedure verifies if the SharedCall extension works properly. |
| **Procedure Steps** | After having executed TPS1, execute S3. |

### 3.4.3. TPS3

| | |
|---|---|
| **Test Procedure Identifier** | TPS3 |
| **Purpose** | This test procedure verifies if the entire Server is properly integrated. In particular we test:<br>• If an appropriate fare is calculated at the end of a Call. |
| **Procedure Steps** | After having executed TPS2, execute SI4 and SI5 |

After having verified TPS3, we can proceed and test the communication part.

At the end, we make an integration test between the Server-side and Client-side using the communication.

# 4. Tools and Test Equipment Required

Supposing that the developer team has used Java language to develop the program, the following tools can be used to performing the test:

- **Jmeter** is used to test if network works, and the performance of the Server in a heavy load situation. We build multiple virtual users that connects to the server, also to understand the maximum load that can be sustained by the Server.

- **Moquito** can be used to for write all mock objects needed (drivers and stubs) to perform various phases of the integration steps.

- **Arquillian** will be used to test if the interaction with the database is correct.

Moreover, manual test can be used to check if all the system works properly and the user experience is good enough.

# 5. Program Stubs and Test Data Required

First, we need that all unit tests has been successfully performed (for example with JUnit).

Following we list all drivers required to perform integration steps.
In the "Functions" columns, we list the function that the driver will call in the corresponding class.

| **Name** | **Functions to be tested** | **Paragraphs** |
|---|---|---|
| Server Driver | Login;<br>Logout;<br>Functions to manage user account;<br>Save call;<br>Add User;<br>Functions to Manage taxi-drivers and manage zones. | CI1 |
| QueueManager Driver | Functions to manage zones queue;<br>Functions to send requests to taxi-drivers | CI2 SI2 |
| User Driver | makeCall | CI4 SI1 |
| User_tSharing Driver | makeSharedCall | CI5 |
| Call Recognizer Driver | recognizeSharedCall;<br>calculateAppropriateFare | CI8 SI4 |
| SharedSet Driver | manageCall;<br>compareCall;<br>createNewSCall | SI3 |

Following we list the various input data required to perform test cases for each function named in the column "Functions to be tested" in the table above.

| **Name** | **Input Data** | **Driver Name** |
|---|---|---|
| Login | - User already registered<br>- User not registered<br>- User already logged in | Server Driver |
| Logout | - User not logged in<br>- User logged in | Server Driver |
| Functions to manage user account | - correct information<br>- incorrect information | Server Driver |
| Save Call | - call details already registered<br>- call details not registered | Server Driver |
| Add User | - not existing user<br>- existing user | Server Driver |
| Functions to Manage taxi-drivers | - request a taxi when at least one taxi-driver is available<br>request a taxi when no taxi-drivers is available<br>- request a taxi and taxi driver refuses<br>- request a taxi and the first taxi driver accepts | Server Driver |
| Functions to Manage zones | - add a not existing zones<br>- add an existing zones<br>- remove a not existing zones<br>- remove an existing zone | Server Driver |
| Functions to manage zones queue | - add a taxi-driver to a zone<br>- remove a taxi-driver to a zone<br>- add a shift to a taxi driver<br>- remove a shift to a taxi driver<br>- add an existing shift to a taxi driver | QueueManager Driver |
| Functions to send requests to taxi-drivers | - taxi driver in service<br>- taxi-driver not in service | QueueManager Driver |
| makeCall | - correct call details<br>- incorrect call details<br>- make a fast-call outside the zone covered by the service<br>- make a fast-call inside the zone covered by the service<br>- make a booked call more than2 hours before the scheduled time<br>- make a call less than 2 hours before the scheduled time | User Driver |
| makeSharedCall | - make a fast and shared call<br>- make a booked and shared call<br>- make a shared call and no compatible existing shared-call<br>- make a shared call and compatible existing shared-call | User_tSharing Driver |

| | | |
|---|---|---|
| recognizeSharedCall | - make a shared call<br>- make a normal call | Call Recognizer Driver |
| calculateAppropriateFare; | - calculate fare for a normal call<br>- calculate fare for a shared call | Call Recognizer Driver |
| manageCall | - modify call with incorrect details<br>- modify call with correct details<br>- modify call 2 hours before the scheduled time<br>- modify call less than 10 minutes before the scheduled time | SharedSet driver |
| compareCall | - existing compatible path<br>- not existing compatible path | SharedSet driver |