Politecnico di Milano

Department of Electronics, Information and Bioengineering

Master Degree course in Computer Science Engineering

Integration Test Plan Document (ITPD)

myTaxiService

Instructor: Prof. Elisabetta Di Nitto

Authors:	*Code:*
Luca Luciano Costanzo	789038
Simone Disabato	852863

Document version 1.0, released on January 16, 2016

Contents

Chapter 1

Introduction

In this chapter the purpose and the scope of the document will be presented in the section 1.1. Then, other useful information are made available, for instance the list of definitions and abbreviations and the reference documents. Finally, in the section 1.4 an overall description of the document structure will be presented.

1.1 Purpose and Scope

The purposes of this document are principally two. The first one is to describe the sequence of myTaxiService component's implementation. The sequence of the implementation of the components described in the Design Document is important for the testing phase.
The second main objective of this document is to define the testing phase of the components' interactions: an accurate description will be provided with the required new components and the tests' definitions, each in a dedicate section.
Finally, in order to accomplish the tests some tools are obviously needed. In this document we will pay particular attention to which tool and how they will be used.

Up to now, we have described the purposes of the document. The global scope of the project follows, as it has been explained in the previous documents, to

make the document easy to understand.
Users, once registered, are able to ask for an immediate ride or to book one of them.

The system provides the user with a complete map of the city and its suburbs within the taxi service is available. The current position of the user is obtained by localization services of the user's smartphone if it's possible, otherwise the user notifies his position directly on the map with a marker or by a searching box. The destination is also chosen either graphically or by a research. The user can view the suggested path and then he must confirm the request.

When a user asks for a ride, the system checks the availability of a taxi driver near the current position, by splitting the city in several areas and using a FIFO (First In First Out) policy to manage the assignment of the ride's driver. The selected driver can accept or decline the ride. In the former case the system informs the user about waiting time, estimated travelling time, prices and cab car-code.

The system gives also the possibility to book a ride with at least two hours in advance. As the user does when he asks for a ride, he selects the desired starting venue and the destination. Afterwards, the system gives a calendar where the customer can choose the date (at most 30 days in advance) and the starting hour. Ten minutes before the meeting time the system starts all the operations described before in order to assign a taxi-driver.

A reservation from the app or the website can be undone until the system confirmation of the availability of a taxi, while a booking can be cancelled at most fifteen minutes before the meeting hour.

After those deadlines the ride is considered bought by the customers and an eventual absence on the established venue forbids other possibilities to book or to take a ride.

1.2 List of Definitions and Abbreviations

Up to now, no definitions or acronyms or abbreviation have been used in the document. Hence, this section is now empty. In future we will provide some important definitions, like the kind od rides, the names of the tools and so on... TO BE CONTINUED

1.3 List of Reference Documents

The reference documents are now listed. Note that, all the documents related on the *myTaxyService* project are written by the same authors of this one, whereas the other documents have a reference of their author when this information is available.

- Software Engineering 2 Project, AA 2015-2016 Assignments 4 - Test plan (available on beep platform only for registered students of Politecnico of Milan);
- The Requirements Analysis and Specification Document (RASD) for *myTaxiService* - v1.2, released on 6th November 2015;
- The Design Document (DD) for *myTaxiService* - v1.0, released on 4th December 2015;
- Mockito reference document, available at
  http://site.mockito.org/mockito/docs/current/org/mockito/Mockito.html or
  http://mockito.github.io/mockito/docs/current/org/mockito/Mockito.html
- Arquillian reference document, available at
  https://docs.jboss.org/author/display/ARQ/Reference+Guide?_sscc=t

1.4 Overall Description

In the chapter 2 the preconditions, the required new components, the temporal development/testing order and the testing strategies are analysed in details. In the chapter 3 each test will be analysed with a brief description and the expected results.

In the chapter 4 a presentation of the used tools we will be provided with the reasons for their use. Finally, in the chapter 5 we will discuss about the special test data or program stubs required for each integration step.

Chapter 2

Integration Strategy

In this chapter the integration strategy will be described. In the section 2.1 the prerequisites for the tests will be presented. The section 2.2 is dedicated to the required components to be integrated in the system to execute some tests. Finally in the sections 2.3 and 2.4 the strategy used to test the integrations will be discussed, paying attention to the order.

2.1 Entry Criteria

The criteria which are required to perform each test are easy. The involved components (and eventually needed program stubs) must be developed and fully tested before executing the integration test.
In addition, for some test, further preconditions may be required. These prerequisites will not be described here, but in the chapter 3, dedicated to test's description.

2.2 Elements to be Integrated

Up to now, no further components need to be integrated to perform integration tests.

2.3 Integration Testing Strategy

We have decided to test our system's interactions with a bottom-up approach. In the section 2.4 it is possible to see the components' order of implementation defined by us with specific criteria. As soon as two components are available, the tests associated to their interactions have to be applied. Before starting the development of the following part of the system, all the tests should be successful.
Obviously, to perform some tests, we need to use a *non-available* component (it has not been implemented yet): to simulate this components we will use some particular program stubs, if any, and they will be described in detail in the chapter 5.

2.4 Sequence of Component/Function Integration

In this section we point out the features related to the order of the components' development. Which is our order of implementation? Why do we need to develop the component *A* before *B*? These two questions are fully discussed from now.

2.4.1 Software Integration Sequence

The "subsystems" of *myTaxiService* are two, both important.
The *Client and User Handler* has the specific role to bind the users' requests to the corresponding services on the system by enqueueing the request on the dispatcher. Then it is able to show the results or to perform the following interactions of the involved service.
To handle these operations it has three components, with the names *Authentication*, *ServiceShow* and *ClientInterface*[1]. In the Figure 2.1 the order of implementation is shown[2].

[1] To have a brief description of each of them read the Design Document.

[2] According to the Design Document the reader may expect to see all the interfaces of the Client and User Handler. To simplify the representation all the interfaces and the classes have

been grouped and inserted into the related component.

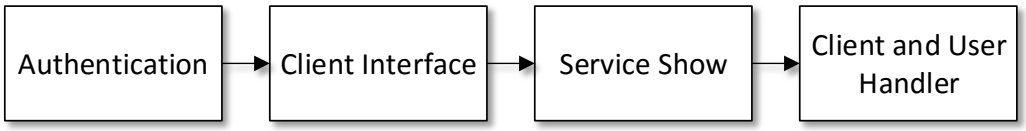

Figure 2.1: Integration Sequence of Client and User Handler

The *Ride Allocator* handles everything that is related to a ride, so the taxy drivers' queues, their assignments and the identification of their positions (by GPS coordinates or an address).
A more precise description is available in the Design Document, as usual, whereas here we are interested only in the implementation order shown in Figure 2.2

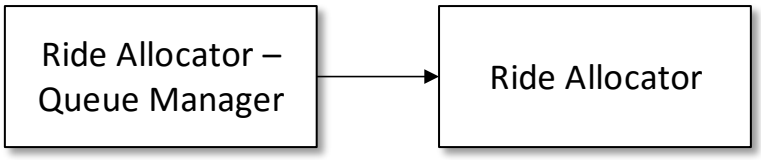

Figure 2.2: Integration Sequence of Ride Allocator

The interactions and the integration inside these two subsystems are not tested in this project phase, but when the single components are implemented. In fact, each component is tested as soon as its first implementation has been completed and until all the tests are successful the development can not be considered done.
This kind of tests are defined in a dedicated document, called *Unit Test Plan Document*, not available now for *myTaxiService*.

2.4.2 Subsystem Integration Sequence

In this section we will present the global interactions between the *myTaxiService*'s core subsystems and we will define all the integration tests and their order.

In the Figure 2.3 we can see the components' implementation order, as we suppose it is better: first of all the DBMS, then, in this order, all related to data checking or handling, to the users, to the security, to the rides and, finally, the dispatcher (the most important component in our system).

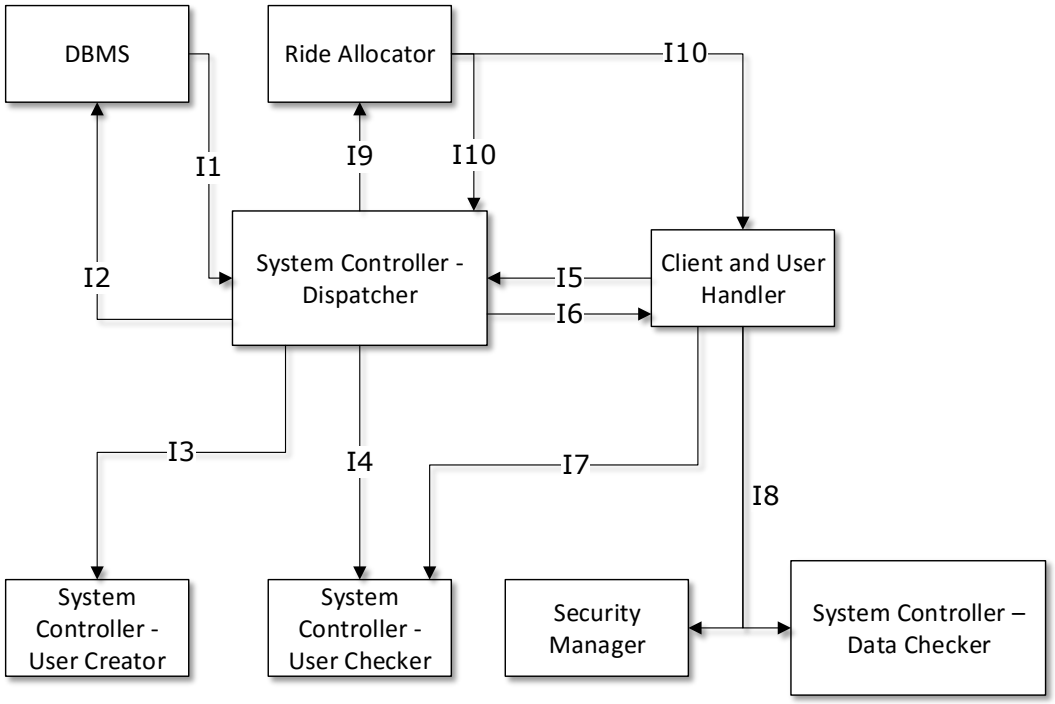

Figure 2.3: Integration Test Order

Integration tests.

ID	Integration test
I1	DBMS → System Controller - Dispatcher
I2	System Controller - Dispatcher → DBMS
I3	System Controller - Dispatcher → User Creator
I4	System Controller - Dispatcher → User Checker
I5	Client and User Handler → System Controller - Dispatcher
I6	System Controller - Dispatcher → Client and User Handler
I7	Client and User Handler → User Checker
I8	Client and User Handler → System Controller - Data Checker , Security Manager
I9	System Controller - Dispatcher → Ride Allocator
I10	Ride Allocator → System Controller - Dispatcher , Client and User Handler

Chapter 3

Individual Steps and Test Description

Chapter 4

Tools and Test Equipment Required

In this chapter we are going to present the tools that will be used to perform the integration test. They will be presented with their name, a link to their reference documents in the chapter 1 and a short explanation concerning the reasons why and the situations where they will be used.
The first one is *Mockito*, available at the website (it is a GitHub repository) `http://mockito.github.io`. This tool is especially useful to design and to implement the program stubs, described in the chapter 5, because it allows to specify predefined *answer* to each method calling with a few lines of code.

The second tool is *Arquillian*, developed to make easy the integration tests. Hence, its use will allow us to perform the integration tests.
Further information about this tool are available at its website, `http://arquillian.org`.

Finally, an important test about the functionalities of *myTaxiService* is about

its capacities and response times during normal or intense executions. This kind of test can be performed by using *Apache JMeter*[1] tool and with a little different client stubs w.r.t. the one defined in the chapter 5 (able to return random, but valid data instead of returning always the same ones). Despite the importance of this kind of tests, they are not argument of this document.

[1]Further information about this tool are available at `http://jmeter.apache.org`.

Chapter 5

Program Stubs and Test Data Required

This chapter is dedicated to a detailed description of each required program stub to the integration tests.

5.1 Program Stubs

The Integration Test of *myTaxyService* requires four program stubs to perform some interactions with the external environment and/or components, like the database.

5.1.1 User Simulator

The User Simulator is a stub able to act as a normal user. It offers the following methods:

- *register*, to simulate a registration with fixed data;
- *login*, to simulate a login with the same credentials defined in the registration phase;
- *profileManagement*, to simulate the management of the user's personal profile. The change is only one for each execution and it is fixed as for the

user's data.

- *zerotimeReservation*, that accepts as parameter GPS coordinates and an address. It simulate a booking of a zerotime ride. The starting position of the ride is given as in the case where the user has activated the GPS and, to be precise, that value is the one passed as parameter.
- *futureReservation*, to perform a future ride booking. For each calling of the method the departure is exactly three hours after the current time, whereas the two addresses are the same passed as parameter to the method[1].

Note that the two remaining functionalities (check the reservations and read the alerts) are not provided by this stub.

5.1.2 Driver Simulator

The Driver Simulator is a stub able to act as a taxy driver. It offers two main methods:

- *accept*, to simulate the acceptance of a System request for a ride's assignment. In each method invocation the action of accepting a ride is performed. An other version of this method, called *randomRide* can either accept or deny a ride, if someone wants to test this specific case.
- *startWaiting*, that accepts a GPS position and with this value it simulates the *Start Waiting Time* functionality.

Note that the working hours handling is not provided.

5.1.3 DataBase Stub

This stub is able to work as a real database for the other components. The methods offered by this stub allow to save and receive data as in a real interaction: in the former case the fake database perform no actions, whereas in the latter case fixed data are returned by each method. We do not point out on each

[1]Both for this method and for the previous one, the addresses are not fixed as other data. The reason for this choice is easy: we can simulate the booking with valid values, depending on the simulated city.

method definition: to have an idea see the DBMS interface into the Design Document and imagine an application of them to the rides' data.

5.1.4 Dispatcher Stub

TO BE WRITTEN

Chapter 6

Other Info

This chapter contains information about the used tools and the hours of work by the members of the working group.

6.1 Working hours

Date	Costanzo's hours	Disabato's hours
2016/1/9	-	1h
2016/1/10	-	1h
2016/1/13	1h	2h
2016/1/14	-	1h
2016/1/15	2h	1h
2016/1/16	1h	-
Total ITPD	4h	6h
Global	68h	70h

6.2 Tools

For this assignment the following tools were used:

- LaTeX and TexStudio editor
- Microsoft Visio 2013