Assignment 4: Integration Test Plan

Matteo Rizzi (789467) & Claudio Scandella (853781)

January 21, 2016

Contents

1 Introduction

1.1 Revision History

This is the first ITP redacted (version 1.0). Therefore there are no previous versions.

1.2 Purpose

The document describes the testing plan for the application functions and components and how they are integrated together in order to test the application code as much as possible in the best way.

1.3 Scope

We will project and implement myTaxiService, which is an application to improve the taxi service of a large city.The service aims to simplify the user's access to the service and to guarantee a fair management of the taxi queues.

1.4 Definitions and Abbreviations

1.4.1 Definitions

- T: we call T the part of our design that has been already subject to integration testing.

1.4.2 Abbreviations

- RASD: Requirement Analysis and Specification Document;
- DD: Design Document;
- ITP: Integration Test Plan;
- UMAV: UserMobileAppView;
- UWAV: UserWebAppViwe;
- TDDAV: TaxiDriverDeviceAppView;
- RR: RequestReceiver;
- RM: RequestManager;
- D: Dispatcher;
- QM: QueueManager;
- PM: PositionManager;
- DM: DataManager;

- AM: AccessManager;
- TM: ReportManager;
- IM: IssueManager.

1.5 References

- RASD version 1.1.pdf;
- Assignment 1 and 2.pdf;
- Design Document.pdf;
- Assignment 4.pdf;

1.6 Overview

This document is divided into six sections:

- Introduction: in this section we will introduce the purpose and the scope of this document and we will make available definitions,abbreviations and references for a better comprehension of the text;
- Integration Strategy: in this section we will describe the Entry Criteria for the integration testing, the elements and components that will be apart of this test and tbe strategy we chose to correctly test our components;
- Individual Steps and Test Description: For each step of the integration process identified above, we will describe the type of tests that will be used to verify that the elements integrated in this step perform as expected;
- Tools and Test Equipment Required: We will identify all tools and test equipment needed to accomplish the integration;
- Program Stubs and Test Data Required:Basing on the testing strategy and test design, we will identify any program stubs or special test data required for each integration step;
- Appendix: in this section we will summarize the tools used to write the document and the hours of work spent by the developers on it.

2 Integration Strategy

2.1 Entry Criteria

The entry criteria are the conditions that must be met in order to proceed with the integration testing of the application elements and components. Those entry criteria are:

- The project is code-complete and so there are no missing parts;
- Verify if test environment is available and ready for use;
- Verify if test tools in the environment are ready for use;
- Verify if testable code is available;
- All elements to be integrated were tested;
- Every test of the different components must have been successful.

2.2 Elements to be integrated

We have to integrate all the different components we identified in the DD. Those elements are the following:

- UMAV;
- UWAV;
- TDDAV;
- Request Receiver;
- Request Manager;
- Queue Manager;
- Position Manager;
- Access Manager;
- Report Manager;
- Issue Manager;
- Data Manager;
- Dispatcher.

In order to know the function of the single components refer to the DD.

2.3 Integration Testing Strategy

The decision is to use the functional groupings method plus a bottom-up strategy for the Integration Test. We decided to initially group our components, represented in the component diagram (see DD), basing on the different functionalities the application will offer. This strategy was chosen because our application's functionalities are quite isolated. Therefore the functionalities tests could still be very satisfactory without testing them with the components that are not directly correlated each other. At the end, when all the groups are correctly tested, a bottom-up integration test will be made in order to control if the different functionality integrate properly each other, even with those that are not directly correlated.

2.4 Sequence of Component/Function Integration

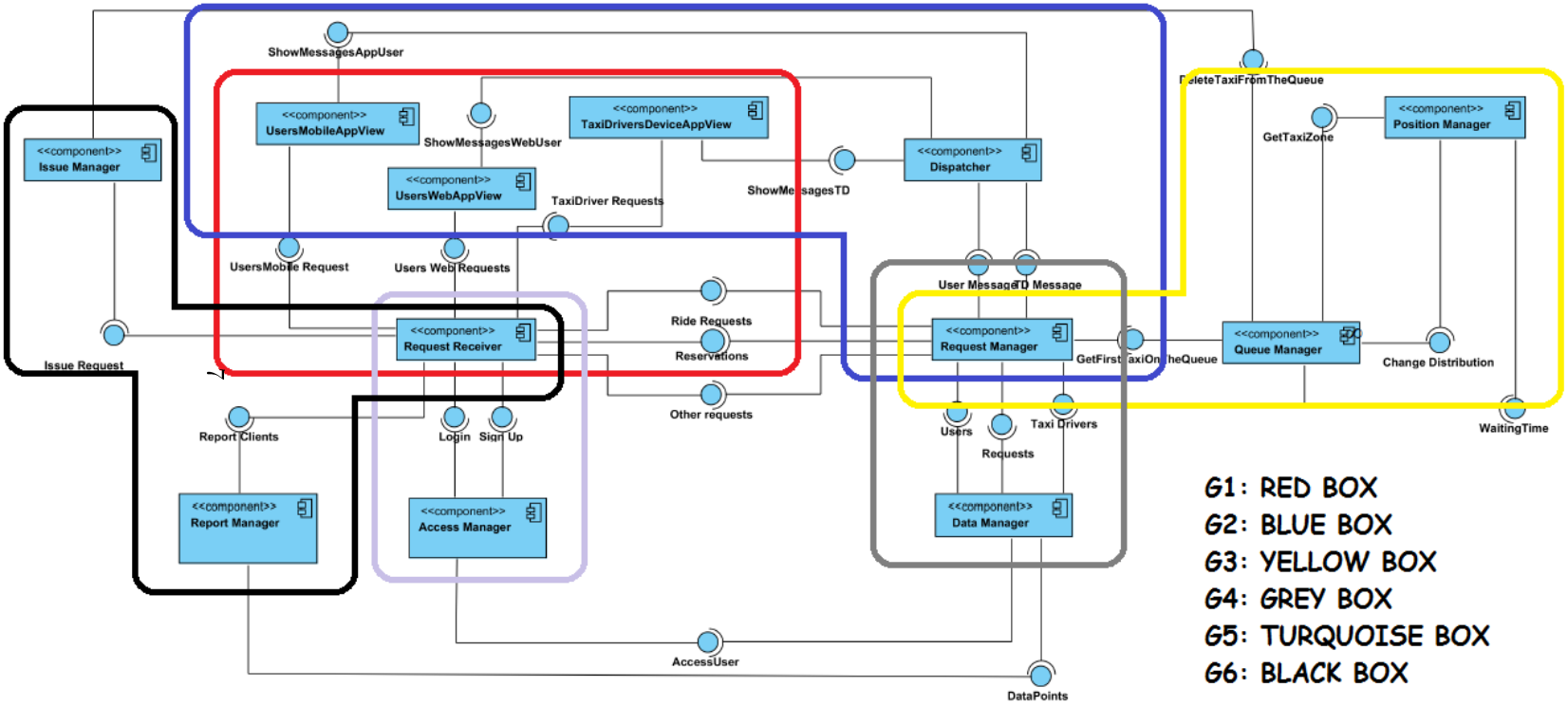
2.4.1 Software Integration Sequence

The application components are divided basing on the functionality they perform as explained in the table that follows:

Functional Group	Components
G1	Request Receiver - UWAV - UMAV - TDDAV
G2	Request Manager - Dispatcher - UWAV - UMAV - TDDAV
G3	Request Manager - Queue Manager - Position Manager
G4	Request Manager - Data Manager
G5	Request Receiver - Access Manager
G6	Request Receiver - Report Manager - Issue Manager

First of all, all groups will be tested individually to check if they do what expected. The order in which groups will be tested is not important.

The picture in the next page show groups in the component view that you can find in Design Document.



G1: RED BOX
G2: BLUE BOX
G3: YELLOW BOX
G4: GREY BOX
G5: TURQUOISE BOX
G6: BLACK BOX

2.4.2 Subsystem Integration Sequence

The decision is to integrate group G1 with group G2 as our first step. Then sequentially a new functional group will be added to the part that has been already tested, basing on the order in which the groups are drafted in the table above. This will continue until all functional groupings are integrated with the others.

3 Individual Steps and Test Description

3.1 Groups test

3.1.1 Group G1

GROUP	G1
ITEMS	UMAV - UWAV - TDDAV - Request Receiver
INPUT	Reception and control of the different commands from the view to the server.
OUTPUT	It checks that the views invoke the expected method in RequestReceiver.
ENVIRONMENTAL NEEDS	Client Driver.
PURPOSE	It controls that UMAV, UWAV and TDDAV calls the method UserMobileRequest, UserWebRequest and TaxiDriverRequest in RequestReceiver, respectively.

3.1.2 Group G2

GROUP	G2
ITEMS	Request Manager - Dispatcher - UWAV - UMAV - TDDAV
INPUT	Generating and sending messages from the server to the different views through the dispatcher.
OUTPUT	It checks that the RequestManager invoke the expected method in Dispatcher which spread messages properly to the views.
ENVIRONMENTAL NEEDS	RequestManager Driver
PURPOSE	It controls that RequestManger calls method UserMessage for messages sent to the users and TDMessage for messages sent to the taxi drivers. Further th Dispatcher must invoke ShowMessageAppUser, ShowMessageWebUser and ShowMessageTD when a message is intended to an app user client, web user client and taxi driver, respectively.

3.1.3 Group G3

GROUP	G3
ITEMS	Request Manager - Queue Manager - Position Manager
INPUT	Management of the queues and position of taxi
OUTPUT	It checks that the RequestManager calls the right method in QueueManager that calls the expected methods in PositionManager. Further the PositionManager must return the exact information to QueueManager.
ENVIRONMENTAL NEEDS	RequestManager Driver
PURPOSE	It check that RequestManager calls the method GetFirstTaxiOnTheQueue in the QueueManager when a taxi is needed. Therefore QueueManager should return the right taxi for the request. QueueManager should invoke GetTaxiZone and ChangeDistribution to get the zone where a request is made and to control the distribution of taxis, respectively. Finally, PositionManager should invoke the WaitingTime method to inform QueueManager of a waiting time.

3.1.4 Group G4

GROUP	G4
ITEMS	Request Manager - Data Manager
INPUT	Data management
OUTPUT	It checks that RequestManager calls expected methods in DataManager
ENVIRONMENTAL NEEDS	RequestManager Driver
PURPOSE	It controls that RequestManager calls Users, Requests and TaxiDrivers in DataManager when an user info, a request info and a taxi driver info is needed to be saved into the DB, respectively.

3.1.5 Group G5

GROUP	G5
ITEMS	Request Receiver - Access Manager
INPUT	Login and SignUp management.
OUTPUT	It checks that the RequestReceiver invoke the proper methods in AccessManager.
ENVIRONMENTAL NEEDS	RequestReceiver Driver
PURPOSE	It controls that RequestReceiver invoke Login method when a login request is sent, and SignUp when a sign up request is sent to the RequestReceiver.

3.1.6 Group G6

GROUP	G6
ITEMS	RequestReceiver - Report Manager - Issue Manager
INPUT	Issues and reports management
OUTPUT	It checks that the RequestReceiver invoke the exact methods in ReportManager ind IssueManager.
ENVIRONMENTAL NEEDS	RequestReceiver Driver
PURPOSE	It checks that RequestReceiver calls ReportClients in ReportManger when a report message is sent to the ReuqestManager. Further it checks than RequestReceiver invoke IssueRequest when a issue message is sent to RequestManager.

3.2 Integrations test

3.2.1 Integration I1

INTEGRATION	G1- G2
ITEMS	UMAV - UWAV - TDDAV - Request Receiver - Request Manager - Dispatcher
INPUT	It manages the entire flow of information exchanged between the client and the server.
OUTPUT	It controls that the Request Receiver, the Request Manager and the Dispatcher calls the right methods in the Request Receiver, the Request Manager, the Dispatcher and the different view, respectively.
ENVIRONMENTAL NEEDS	Client Driver.
PURPOSE	It controls that the Request Receiver calls the three different ways of communication with the Request Manager basing on the type of message sent by the client. The Request Manager , when the messages are ready to be sent to the clients, must send them to the Dispatcher, in order to broadcast them to the different clients. The Dispatcher must, then, broadcast the messages in the right way; so, for example, the messages sent to taxi drivers, will be sent calling the right interface method with TDDAV and in the same way (with the correct method) with the other views.
PROCEDURE STEPS	Add G3 to the current integration.

3.2.2 Integration I2

INTEGRATION	T-G3
ITEMS	UMAV - UWAV - TDDAV - RR - RM - D - QM - PM
INPUT	A new functionality is added. The queue management function and the extrapolation of the taxi and client position on the map
OUTPUT	It checks the correct execution of the work flow between Request Manager and Queue Manager and betwwen the Queue Manager and the Position Manager.
ENVIRONMENTAL NEEDS	Client Driver. The previous integration must have been succesful.
PURPOSE	It controls that the Request Manager, when it receives any kind of request, elaborates it and calls the correct function of taxi extrapolation from the queue, with the help of the Position Manager.
PROCEDURE STEPS	Add G4 to the current integration

3.2.3 Integration I3

INTEGRATION	T-G4
ITEMS	UMAV - UWAV - TDDAV - RR - RM - D - QM - PM - DM
INPUT	Functionality to access, throw the RM-DM interface, to the DB is added
OUTPUT	It controls the correct call of the interface methods between the Request Manager and the Data Manager for appropriate commands received from clients.
ENVIRONMENTAL NEEDS	Client Driver. Test account in DB. The previous integration must have been succesful.
PURPOSE	To check that, when the client send a request that is forwarded to the Request Manager, it will interface with the right methods to the Data Manager and the data will correctly be saved in the DB
PROCEDURE STEPS	Add G5 to the current integration

3.2.4 Integration I4

INTEGRATION	T-G5
ITEMS	UMAV - UWAV - TDDAV - RR - RM - D - QM - PM - DM - AM
INPUT	The client Login and SignUp functions are added.
OUTPUT	To check the correct call to the Access Manager methods from the Request Receiver. It also checks the Data Manager correct function when called by the Access Manager
ENVIRONMENTAL NEEDS	Client Driver. The previous integration must have been succesful.
PURPOSE	To verify that when the client makes a Login request it will not be already logged in and that he really is a subscriber in the DB. To control also that when a signup is requested it does not cause conflicts with data already stored in the DB
PROCEDURE STEPS	Add G6 to the current integration

3.2.5 Integration I5

INTEGRATION	T-G6
ITEMS	UMAV - UWAV - TDDAV - RR - RM - D - QM - PM - DM - AM - TM - IM
INPUT	The report and issue management functions are added
OUTPUT	To check the correct invocation of methods between: • Request Receiver and Issue Manager • Issue Manager and Queue Manager • Request Receiver and Report Manager • Report Manager and Access Manager
ENVIRONMENTAL NEEDS	Client Driver. The previous integration must have been succesful.
PURPOSE	To control that the Report Manager calls the right method to interface with the Data Manager when a Report request is made available by the Request Receiver. To verify that when an Issue request arrives from a Taxi Driver, this will be directed to the Issue Manager with the right method and that the Issue Manager calls the right method of the Queue Manager in order to interface with him.
PROCEDURE STEPS	none

4 Tools and Test Equipment Required

Basing on the way the testing phase has been designed, during the tests of functionality and different integrations for checking the correct working of the classes, it is necessary to use stubs: they are classes that simulates other classes behaviors. "Mokito" tool will be used to create class stubs.

In group G1, the test need a RequestManager to make RequestReceiver send some messages.

In group G3 and G4 a Dispatcher stub is needed to simulate the sending of messages.

In group G5, DataManager stub is required to simulate di access of a user to the application.

In group G6, DataManager and QueueManager are needed to simulate the updating of report points for first one, and delete a taxi for the latter.

In the integration I3, the test needs a test account in the DB to make DM perform insertion of information in it.

In the integration I4, where the DataManager is added, will be used manual testing to create a test account and then try to ricreate it with the same data. The application doesn't should allow the creation a second time with same data.

5 Program Stubs and Test Data Required

We decided to use some stubs in order to correctly simulate some classes of the project. We list now the stubs we decided to use dividing them basing on the functionality and integration testing phase in which they're necessary. They're reported in the following tables:

Group	Stubs and Equipment
G1	RequestManager to make the RequestReceiver send some messages when a user request is done
G2	none
G3	Dispatcher to simulate the sending of messages to users
G4	• Dispatcher to simulate the sending of messages to users • Test account to allow DM work properly
G5	DataManager to simulate the updating of the user state (logged in or not)
G6	• DataManager to simulate the updating of report points to the user that does the irregularity • QueueManager to simulate the deletion of a taxi in a queue when a problem is sent from a taxi driver

Integration	Stubs and Equipment
I1	• Queue Manager stub to return a generic taxi without search in queues; • Data Manager stub to simulate an insertion in DB.
I2	Data Manager stub to simulate an insertion in DB.
I3	Test account to allow DM work properly
I4	Test account to allow DM work properly
I5	Test account to allow DM work properly

6 Appendix

6.1 Tools used

- LYX: in order to redact the ITP document.

6.2 Hours of Work

This is the amount of time each developer worked in order to redact this document:

- Rizzi Matteo: 12 hours;
- Scandella Claudio: 12 hours.