POLITECNICO
MILANO 1863

Software Engineering 2 Project

MyTaxiService

PART 4:
Test Plan Document

Valeria Deciano 858479
Fabio Calabretta 852717

21/1/2016 v 1.0

SUMMARY

1. Introduction

1.1 Revision History

Version 1.0 : actual version.

1.2 Purpose and Scope

This document is used to explain how to implement the integration tests. These are tests that show abnormal behaviour related to the integration of multiple applications. The document, addressed to those who develop the tests, must explain what to test and must provide a testing plan that determines the order in which various interactions should be tested. Are also established what tools are needed for testing.

1.3 List of Definitions and Abbreviations

RASD: Requirements Analysis and Specification Document
DD: Design Document

1.4 List of Reference Documents

Documents to which we refer are:
- RASD
- DD
- Project Description

2. Integration Strategy

2.1 Entry Criteria

Before performing integration tests, it is necessary that each module, employed in the test, has been already tested through modular testing, to ensure that there are no errors in the operation before being integrated with the rest of the system.

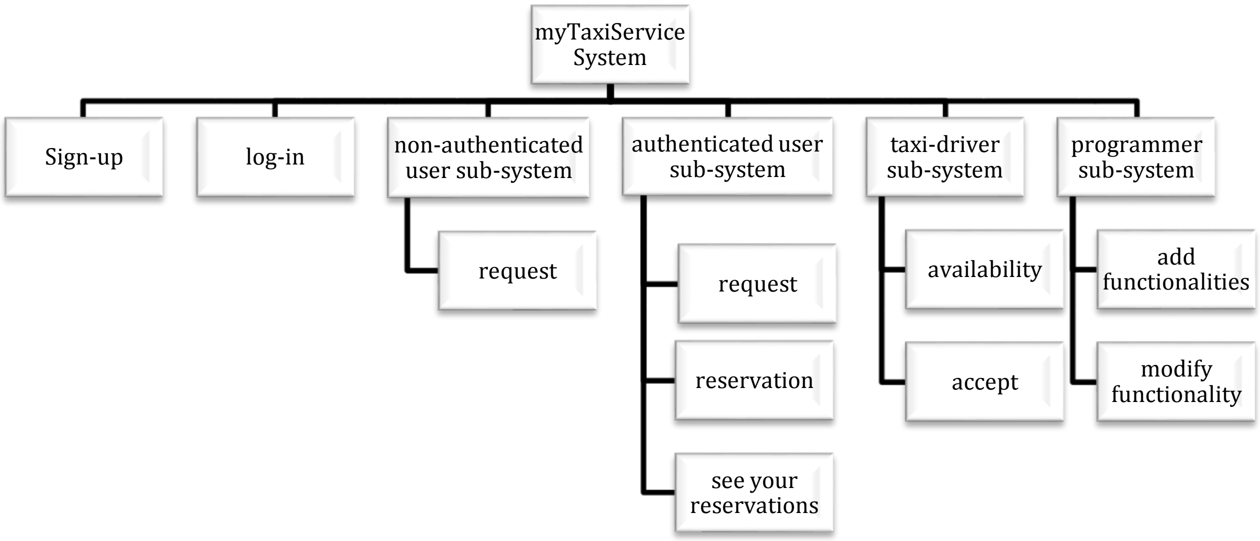
It is also necessary that, before the execution of an integration testing, a set of suitable data is generated to be put in input.

Finally, before the tests, the components involved must be developed. This follows that the documents RASD and DD should have been made, but also the Test Plan.

2.2 Elements to be Integrated

As we define in the DD, section 2.3, the components to be integrated are:

- Sign-up sub-system
- Log-in sub-system
- Non authenticated user sub-system
  - Request sub-system
- Authenticated user sub-system
  - Request sub-system
  - Reservation sub-system
  - See Your Reservations sub-system
- Taxi-driver sub-system
  - Availability sub-system
  - Accept sub-system
- Programmer sub-system
  - Add functionality
  - Modify functionality



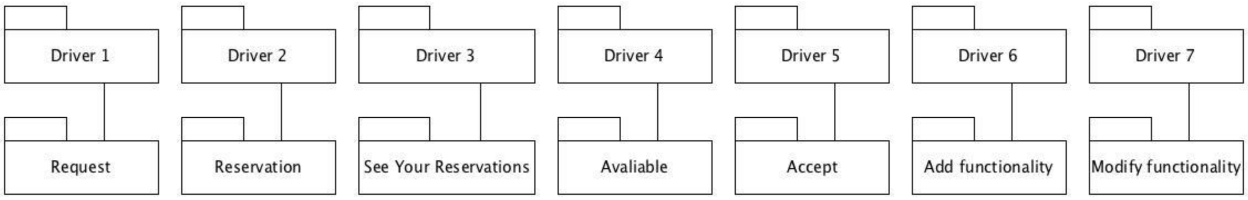
2.3 Integration Testing Strategy

We choose to use a bottom-up approach because in the sub-systems identification phase we chose a top down approach. The structure of the system is a tree and the easiest way is to follow it from the leaves to the root.

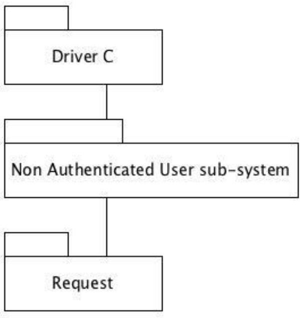
2.4 Sequence of Component/Function Integration

As we said in section 2.2, we chose a bottom-up approach, so after testing every subsystem with unit tests then we continue with integration tests as follows.

Before doing integration testing, each subsystem needs to be tested with a driver that simulates the behavior of the other subsystems, with the aim of seek errors. The description of the various drivers following in chapter 5

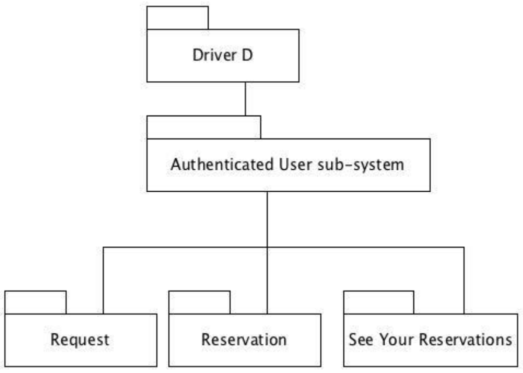


After we can do an integration test between “Request” and “Non Authenticated User Sub-System”:



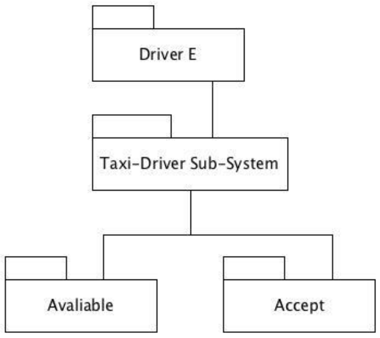
IT1: Request → Non Authenticated user sub-system

Thus we can do an integration test between “Request”, “Reservation” and “See Your reservation” with “Authenticated User sub-system”:



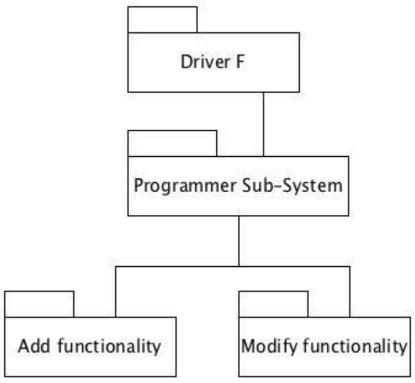
IT2: Request, Reservation, See Your reservation → Authenticated User sub-system

Another integration test is between “Available”, “Accept” and “Taxi-Driver sub-system”:



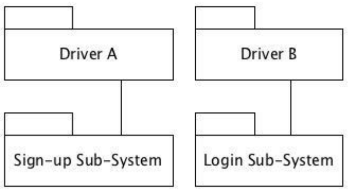
IT3: Available, Accept → Taxi-Driver sub-system

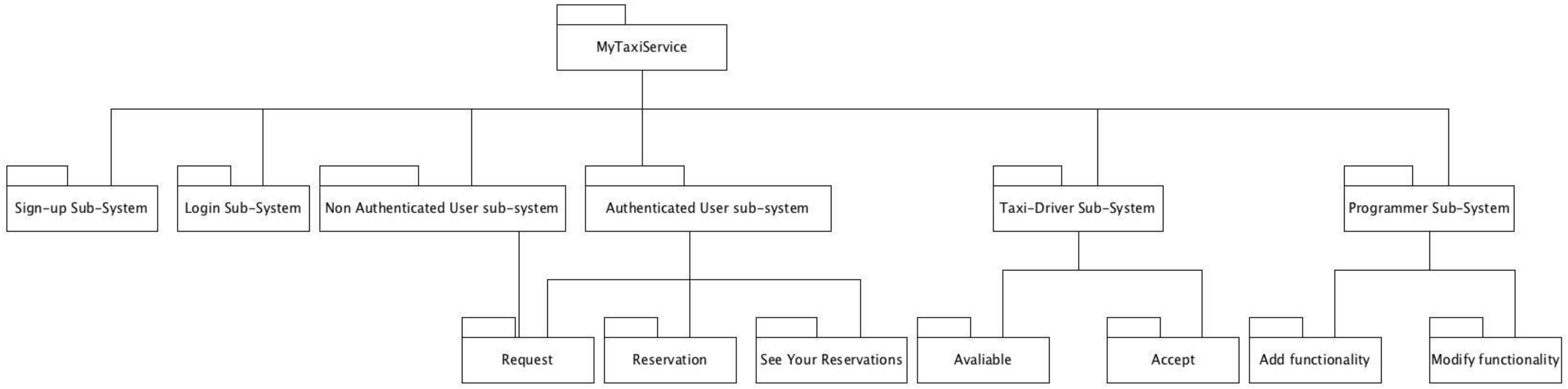
Now, we have an integration Test between “Add Functionality”, ”Modify Functionality” and “Programmer sub-system”:



IT4: Add Functionality, Modify Functionality → Programmer sub-system

The last integration test is between “Sign-up sub-system”, “Log-in sub-system”, “Non authenticated user sub-system”, “Authenticated user sub-system”, “Taxi-driver sub-system”, “Programmer sub-system” and “MyTaxiService System”





IT5: Sign-up sub-system, Log-in sub-system, Non authenticated user sub-system, Authenticated user sub-system, Taxi-driver sub-system, Programmer sub-system → MyTaxiService System

3. Individual Steps and Test Description

3.1 Integration Test case IT1

Test Case Identifier	IT1- T1
Test Item(s)	Request → Non Authenticated user sub-system
Input Specification	Request made by non authenticated user
Output Specification	Request data in Database related with no authenticated user
Environmental Need	Request Unit test and Non Authenticated user sub-system Unit test succeeded

3.2 Integration Test case IT2

Test Case Identifier	IT2-T1
Test Item(s)	Request, Reservation , See your Reservations → Authenticated user sub-system
Input Specification	Normal Request input
Output Specification	The request data aren't in Database as reservation Data
Environmental Need	Request Unit Test, Reservation Unit test, See Your Reservations Unit test and and Authenticated user sub-system test succeeded

Test Case Identifier	IT2-T2
Test Item(s)	Request, Reservation , See your Reservations → Authenticated user sub-system
Input Specification	Normal Reservation input
Output Specification	The reservation data aren't in database as Request data
Environmental Need	Request Unit Test, Reservation Unit test, See Your Reservations Unit test and and Authenticated user sub-system test succeeded

Test Case Identifier	IT2-T3
Test Item(s)	Request, Reservation , See your Reservations → Authenticated user sub-system
Input Specification	Normal Reservation input
Output Specification	The reservation data are available in See Your Reservation
Environmental Need	Request Unit Test, Reservation Unit test, See Your Reservations Unit test and and Authenticated user sub-system test succeeded

Test Case Identifier	IT2-T4
Test Item(s)	Request, Reservation , See your Reservations → Authenticated user sub-system
Input Specification	Normal Request input
Output Specification	The Request data aren't in See Your Reservation
Environmental Need	Request Unit Test, Reservation Unit test, See Your Reservations Unit test and and Authenticated user sub-system test succeeded

Test Case Identifier	IT2-T5
Test Item(s)	Request, Reservation , See your Reservations → Authenticated user sub-system
Input Specification	Normal Request input
Output Specification	The request aren't accept if there is already a booking for the same moment, by the same user.
Environmental Need	Request Unit Test, Reservation Unit test, See Your Reservations Unit test and and Authenticated user sub-system test succeeded

3.3 Integration Test case IT3

Test Case Identifier	IT3-T1
Test Item(s)	Availability, Accept → Taxi-Driver sub-system
Input Specification	No Communication of Availability
Output Specification	No possibility to accept a route
Environmental Need	Availability Unit test , Accept Unit test and Taxi-Driver sub-system Unit test succeeded

3.4 Integration Test case IT4

Test Case Identifier	IT4-T1
Test Item(s)	Add Functionality, Modify Functionality → Programmer sub-system
Input Specification	Add a new functionality
Output Specification	It must be possible to modify the new functionality
Environmental Need	Add Functionality Unit test , Modify Functionality Unit test and Programmer sub-system Unit test succeeded

3.5 Integration Test case IT5

Test Case Identifier	IT5-T1
Test Item(s)	Sign-up sub-system, Log-in sub-system, Non authenticated user sub-system, Authenticated user sub-system, Taxi-driver sub-system, Programmer sub-system → MyTaxiService System
Input Specification	Sign-up form submitted
Output Specification	The user must have the possibility to log-in successfully
Environmental Need	Sign-Up Sub-System Unit test, Login Sub-System Unit test, MyTaxiService System Unit test, IT1, IT2, IT3 AND IT4 succeeded

Test Case Identifier	IT5-T2
Test Item(s)	Sign-up sub-system, Log-in sub-system, Non authenticated user sub-system, Authenticated user sub-system, Taxi-driver sub-system, Programmer sub-system → MyTaxiService System
Input Specification	Login form submitted
Output Specification	The Home page related to the kind of user must be shown to him
Environmental Need	Sign-Up Sub-System Unit test, Login Sub-System Unit test, MyTaxiService System Unit test, IT1, IT2, IT3 AND IT4 succeeded

3.6 Test Procedures Descriptions

Test Procedure Identifier	TP1
Purpose	This procedure verifies if the system is able to manage the input the client submit and the output he expects
Procedure Steps	IT2-T3, IT2-T4, IT5-T2

Test Procedure Identifier	TP2
Purpose	This procedure verifies if the system is able to manage correctly the client input with the database
Procedure Steps	IT1, IT2-T1, IT2-T2, IT2-T5, IT5-T1, IT5-T2

Test Procedure Identifier	TP3
Purpose	This procedure verifies if the system is able to manage the part of them in a correct way, depending on the client input given.
Procedure Steps	IT3, IT4, IT5-T1

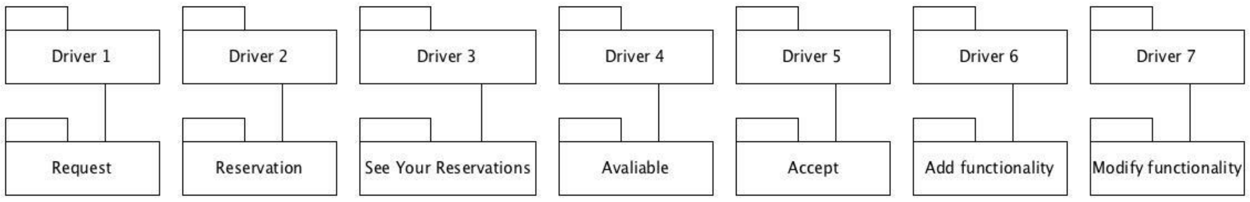
4. Tools and Test Equipment Required

We decide to use these tools for the integration tests: Arquillan and jUnit. Arquillan is particularly adept to test the integration of the subsystems and their interfaces and also for the connections with databases.

5. Program Stubs and Test Data Required

In our test plan we decided to use a bottom-up approach, this involves the definition of some drivers, which are pieces of code that are written to simulate the behaviour of the other modules not yet integrated.

In section 2.4 we identified some drivers that now describe in detail.



Driver 1: This Driver represents the Taxi-Driver and if he accepts the request updates the taxi code and the request state.

Driver 2: This driver simulates the behaviour of a taxi driver who, a few minutes before the route, accept it, so the state of the reservation changes and it is associated with the taxi driver code.

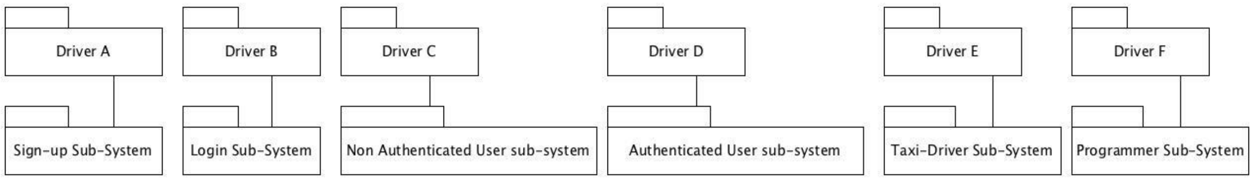
Driver 3: this driver is equal to the Driver 2

Driver 4: This driver manages the taxi queue and when the taxi driver communicates his availability, it put him in the end of the queue related to the zone he is.

Driver 5: This driver manages the relationship with the request/reservation submitted by a user. It assigns a request/reservation to the first taxi driver in the queue relatives to the request/reservation zone.

Driver 6: This driver isn't really necessary because we don't have to check this subsystem with other part of the system, infect, the only thing this subsystem do is to add the functionality code submitted by the programmer.

Driver 7: This driver simulates the functionality submission by a programmer and we want that after this, the new functionality code is visible and it is possible to modify it.



Driver A: This driver isn't really necessary because the only thing this subsystem does is to add User in the database.

Driver B: This driver isn't really necessary because the only thing this subsystem does is to keep User information by database to recognize users.

Driver C: This driver relates the request made by a non-authenticated user with the taxi driver who accepts the route and it update the request information.

Driver D: This driver relates the request/reservation made by an authenticated user with the taxi driver who accepts the route and it update the request/reservation information.

Driver E: This driver relates the taxi driver with the request/reservation made by users, managing the queues.

Driver F: This driver is not really necessary because the Programmer Subsystem has no interaction with the other parts of the system.