TAXI SERVICE

ITPD

Integration Test Plan Document

version 1.1

Rota Diego, 841344, 5 hours

Montalto Simone, 841359, 5 hours

Politecnico di Milano, A.A. 2015-2016

Software Engineering 2 – Prof.ssa Mirandola Raffaela

Table of contents

1. Introduction

1.1 Revision History

v1.1: initial version.

1.2 Purpose and Scope

This document describes how components and subsystem must be integrated during the integration test phase, in order to discover problems and bugs to correct before the final release of the system. This document must be read by engineers that work on the testing of the system to evaluate if all the components work well together. This must be considered a guideline to follow during the integration test. This may help to avoid part of the system that are not tested correctly together.

1.3 List of Definitions and Abbreviations

- **Guest**: it is an user that is not yet registered;
- **Customer**: it is an user that is registered and correspond to the passenger;
- **Driver**: it is an user that is registered and correspond to the taxi driver;
- **External Developer**: it is an user that is registered and can only require the API of the system;
- **User**: it could be a Guest, Customer, Driver or External Developer.
- **Vehicle**: correspond to the taxi car that is driven by a Driver;
- **Request**: it is a generic reservation made by a Customer;
- **Immediate Request**: it is a specific Request, and it is made by the Customer to require a taxi as soon as possible;
- **Booked Request**: it is a specific Request, and it is made by the Customer to book a taxi for a specific hour, origin and destination address;
- **Zone**: it indicates a specific area of the city that includes only one queue.

1.4 List of reference Documents

This is the list of the reference documents:

- Taxi service project specifications;
- Rota-Montalto RASD v1.2;
- Rota-Montalto DD v1.1
- JUnit documentation: http://junit.org/javadoc/latest/index.html
- JMeter user manual: http://jmeter.apache.org/usermanual/index.html

2. Integration Strategy

2.1 Entry Criteria

Before to start the integration test, there are some criteria that must be met. An unit test is performed for each method, class and package. This is performed in order to avoid problems during the next phase that are correlated to a bad code implementation. To the team that is involved in the integration test these items are delivered:

- The Design Document where are explained the components involved in the test;
- Unit Test reports concerning the components involved;
- The JavaDOC concerning the components involved;
- Input data used to accomplish the integration test;
- Drivers for the strategy used (described below).

2.2 Elements to be Integrated

To have an overview of the components involved in the integration test, please see the Design Document. Here are briefly show the elements in order to have an high level vision:

- Subsystem: System
  - RequestManagement
  - ZoneManagement
- Subsystem: Database
  - UserRegistry
  - Zone
  - ImmediateRequest
  - BookedRequest
- Subsystem: DriverClient
  - AvailabilityForm
  - RequestAnswer
  - Authentication
- Subsystem: CustomerClient
  - BookedRequestForm
  - ImmediateRequestForm
  - Authentication

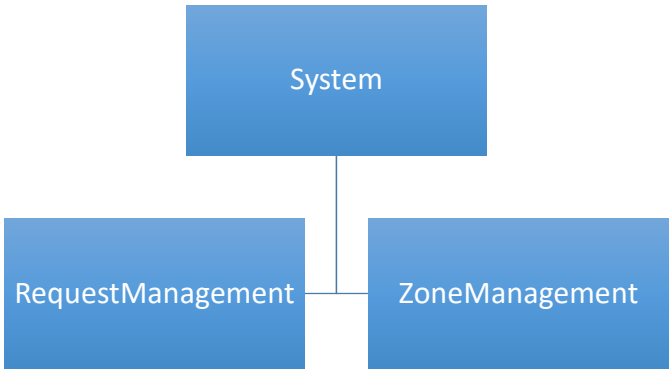
2.3 Integration Test Strategy

For the Integration Test, the strategy that will be used is the Bottom-Up Strategy. To accomplish this kind of strategy, each component is taken singularly and successively will be integrated with another component until all the components are integrated consistently and without errors. To proceed, Drivers are used. A driver is a program that accept data as input and uses it in order to accomplish the integrated test. Each time that two components are integrated, a new driver is used until the system is completely integrated. We have choose this strategy because, starting from single components that have been tested singularly, we'll proceed adding other components that singularly work well.

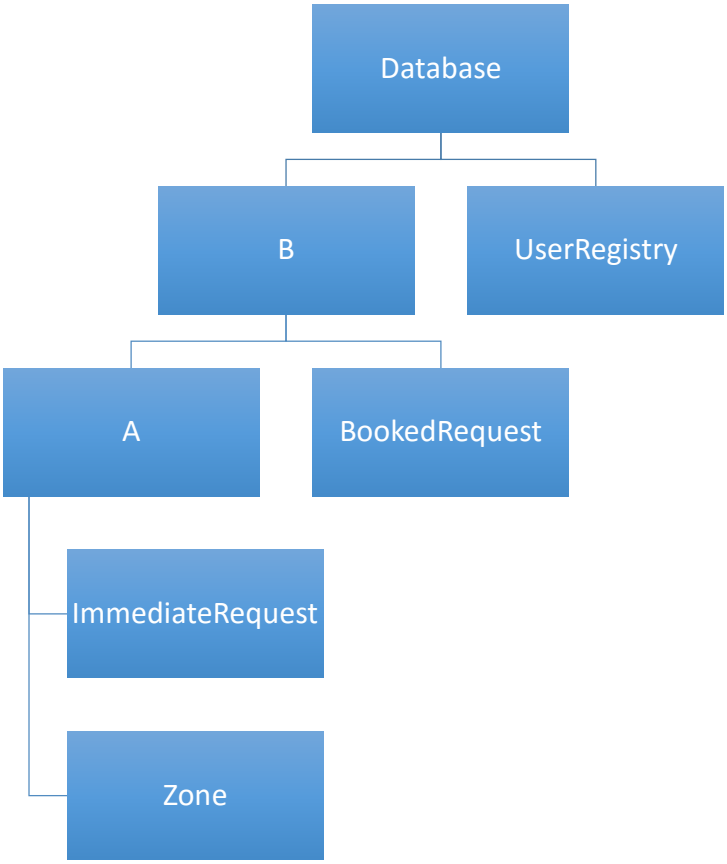
2.4 Sequence of Component/Function Integration

2.4.1 Software Integration Sequence

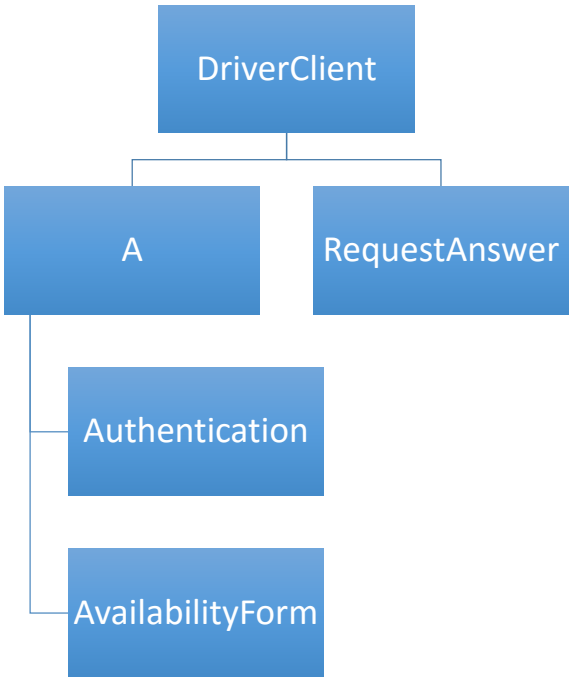
In order to explain the sequence in which the software components will be integrated, a schema is proposed. The schema presents different trees. Each tree represents a subsystem that has the different components as leaves, and starting from bottom to the top, is show the integration sequence.



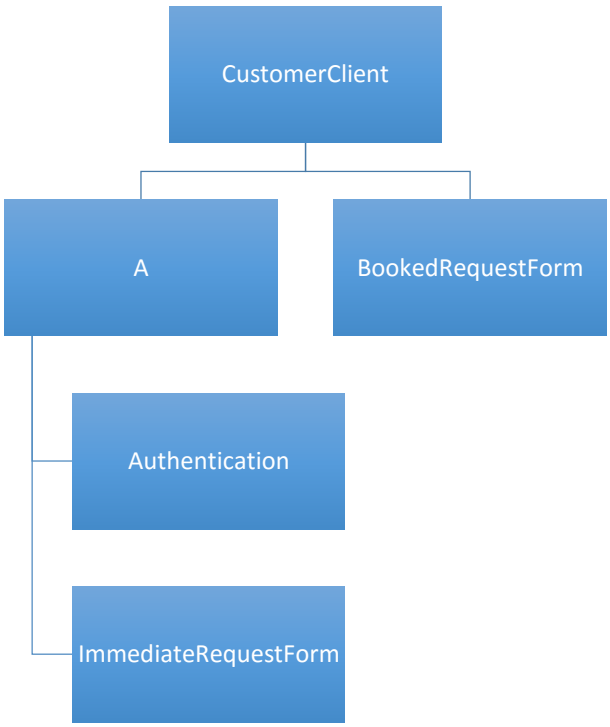
Subsystem: System



Subsystem: Database



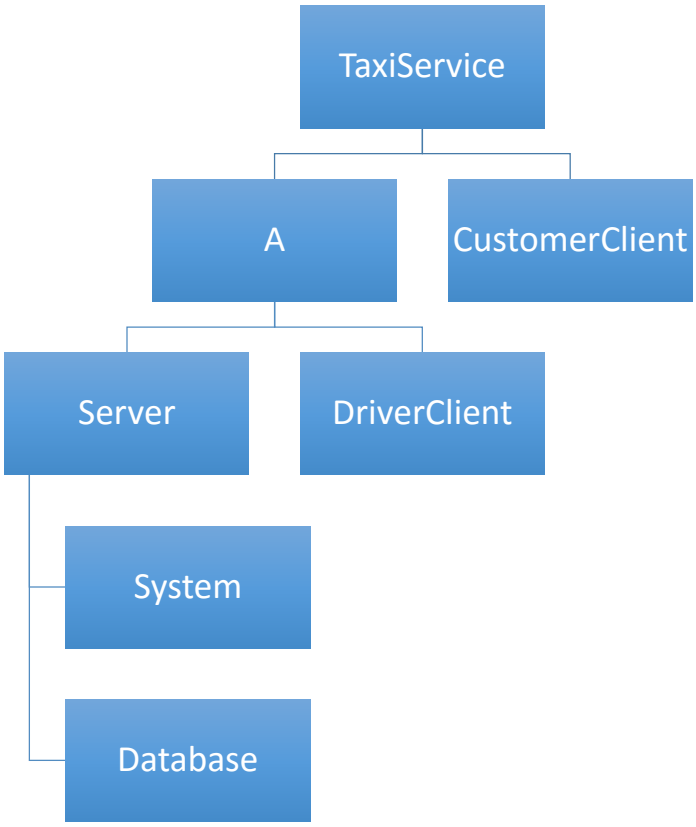
Subsystem: DriverClient



Subsystem: CustomerClient

2.4.2 Subsystem Integration Sequence

As in the previous section, a schema is proposed in order to show the integration sequence of the four subsystem.



Entire System

3. Individual Steps and Test Description

In this section are show the integration test that must be performed in order to verify if the output are as expected. We follow the tree of the section 2.4.2 of this document to perform the integration test design.

3.1 Integration Test System-Database

In this test a RequestManagement is performed by **System** to respond to an ImmediateRequest.

Test case identifier	T1
Test item(s)	System, Database.
Input specification	System manages a new ImmediateRequest through RequestManagement.
Output specification	ImmediateRequest database must be updated
Procedure steps	This test must verify if, when a new ImmediateRequest is managed by System, some tables of the database are updated consistently.
Environment needs	-

In this test a new RequestManagement is performed by **System** to respond to a BookedRequest.

Test case identifier	T2
Test item(s)	System, Database.
Input specification	System manages a new BookedRequest through RequestManagement.
Output specification	BookedRequest database must be updated
Procedure steps	This test must verify if, when a new BookedRequest is managed by System, some tables of the database are updated consistently.
Environment needs	-

In this test ZoneManagement is performed by **System** after that RequestManagement manages a Booked or an Immediate Request.

Test case identifier	T3
Test item(s)	System, Database.
Input specification	ZoneManagement is called by RequestManagement after that it manages a new Immediate or Booked Request
Output specification	Zone database must be updated
Procedure steps	This test must verify if, when ZoneManagement is managed by System, some tables of the database are updated consistently.
Environment needs	T1, T2

3.2 Integration Test Server-Driver Client

In this test, AvailabilityForm is performed by the subsystem DriverClient. The System responds managing the request with ZoneManagement, that must update the Zone database.

Test case identifier	T4

Test item(s)	System, Database, DriverClient.
Input specification	A Driver gives its availability.
Output specification	Zone database must be updated.
Procedure steps	This test must verify if, when ZoneManagement is managed by System after a Driver gives its availability, some tables of the database are updated consistently.
Environment needs	-

In this test, RequestAnswer is executed when a Driver answer to a Client Request. The System executes RequestManagement and the database is updated consistently with the answer.

Test case identifier	T5
Test item(s)	System, Database, DriverClient.
Input specification	A Driver answer to a Request (Immediate or Booked).
Output specification	BookedRequest or ImmediateRequest database must be updated.
Procedure steps	This test must verify if, when RequestManagement is managed by System after a Driver swer to a request, some tables of the database are updated consistently.
Environment needs	T1, T2

3.3 Integration Test Server-DriverClient-CustomerClient

In this test, ImmediateRequestForm is performed by the subsystem **CustomerClient** to request an ImmediateRequest. A Driver must answer. RequestAnswer of the subsystem DriverClient is executed to perform this action, after that the System has managed the request using RequestManagement.

Test case identifier	T6
Test item(s)	System, Database, CustomerClient, DriverClient.
Input specification	A client perform an Immediate Request. Driver answer to the request.
Output specification	ImmediateRequest database must be updated.
Procedure steps	This test must verify if, when a new ImmediateRequestform is executed by CustomerClient, the System manages the request using RequestManagement. On the DriverClient must be executed RequestAnswer. Some tables of the database are updated consistently.
Environment needs	T1, T3, T5

4. Tools and Test Equipment Required

The first step before to start with the integration test is to test the Server code (the subsystem System and the subsystem Database) written in Java (methods, classes and packages). To accomplish it, JUnit is used for the unit test. JUnit have to cover at least the 85% of the methods for each class. For the Android application, JUnit is used for the unit test and internal beta tester help with manual testing. For the iOS application, the Xcode environment is used for unit test and also internal beta tester help with manual testing. For the web application, manual testing is performed.

We use JUnit and Xcode because are tools that are integrated in the environment tool used to develop the system.

For the integration test, JMeter tool is used. JMeter is useful because we can simulate more users that accomplish more actions simultaneously, and this could help to verify if all the components works fine together with an high rate of users. For this reason we can test two important aspects:

1. If the server holds an high rate of users;
2. If the system responds correctly when more than one user uses different parts of Taxi Driver System.

5. Program Stubs and Test Data Required

In order to produce useful result during the integration test, these are stubs and data required that we generate automatically:

- We use a random function that generates predefined response for driver availability. This function is used when, in the integration test, must be tested RequestAnswer module in DriverClientSubsystem.
- We have generated a list with a lot of addresses of Milan metropolitan area, date and hour that are used to generate new Immediate or Booked Request. This list is used, particularly, when, during the integration test, ImmediateRequestForm and BookedRequestForm of the subsystem CustomerClient and RequestManagement and ZoneManagement of the subsystem System are tested.
- UserRegistry database is populated by users generated casually by a function (name, surname and other useful information used during the test).