My Taxi Service

Integration Test Plan Document

Authors:

Andrea DONATI {andrea4.donati@mail.polimi.it}

Gabriele CARASSALE {gabriele.carassale@mail.polimi.it}

Manuel DELEO {manuel.deleo@mail.polimi.it}

Prof: Elisabetta Di Nitto

Milan, January 21, 2016

Contents

1 Introduction

1.1 Revision History

Versions:

- 1.0: Creation of the document

1.2 Purpose and Scope

This is the Integration Test Plan Document (ITPD) relative to the myTaxiService system. Its purpose is to describe how to perform the integration testing of the components of the system. The strategy will be based on previous statements present in the Design Document and in the Requirement Analysis and Specification Document of the project.

1.3 Abbreviations

- **In:** n-th test case
- **Sn:** n-th stub

1.4 List of Reference Documents

- Requirements Analysis and Specification Document (RASD) of myTaxiService
- Design Document (DD) of myTaxiService
- Assignments 1 and 2 (RASD and DD), i.e. the project specifications
- Assignment 4 - integration test plan.pdf, i.e. the structure of the ITPD
- The Integration Test Plan Example document: Integration Test Plan Example.pdf

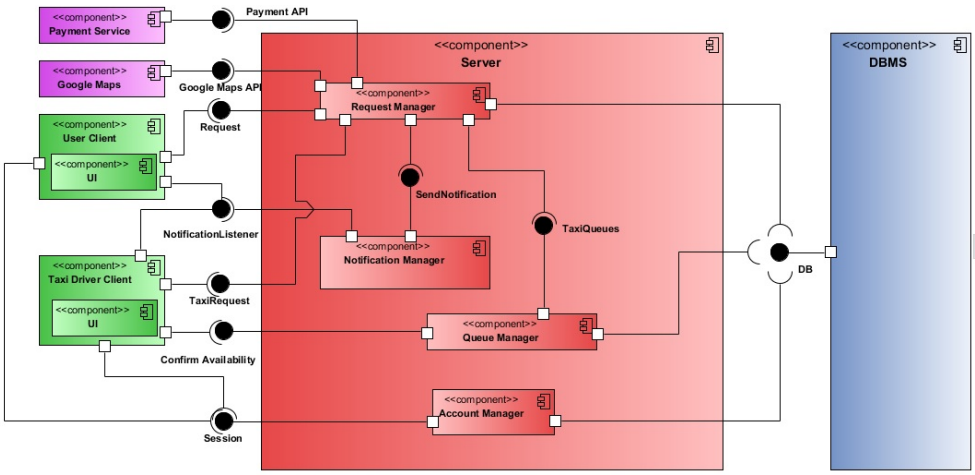
2 Integration Strategy

2.1 Entry Criteria

Before starting with the Integration Test of the system, the code of every single component has to be inspected, either manually or with automated static analysis. Found faults have to be corrected before going on with the Test Plan. In addition, every component has to be unit-tested, except for the external services, which have already been tested by their producers.
Components that must be analyzed and unit-tested, in order to be integrated in the system, will be presented in the next section.

2.2 Elements to be Integrated

As shown in the Design Document, the system is composed of the following components:



The main subsystems are the following:

- DBMS
- Business Logic
- External Services
- Client

The main components, which must be integrated, are the following:

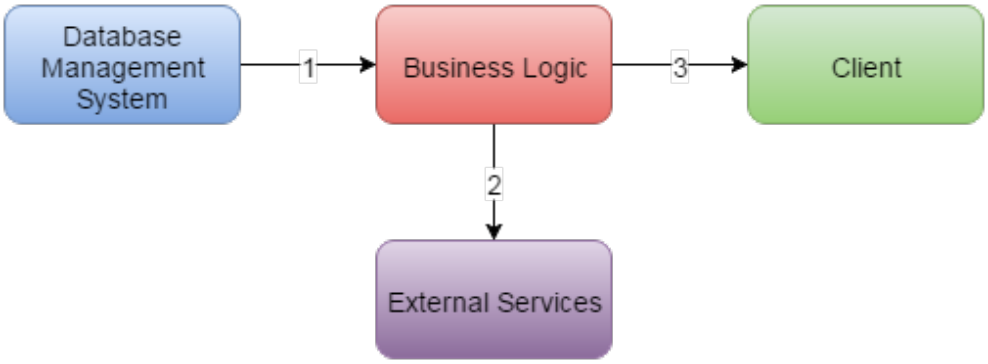
- DBMS
- Account Manager
- Request Manager
- Queue Manager
- Notification Manager

- Google Maps API
- Payment Service
- Taxi Driver and User clients

2.3 Integration Testing Strategy

Said that the components of our system are already developed and unit-tested, we'll use a slightly modified bottom-up integration strategy in order to integrate them into the system. Using the bottom-up approach, we will start integrating the components that are already developed and that need only already tested components.
Moreover, we will start integrating the components from the most independent ones to the least ones. This strategy brings us some important advantages. In this way we can reduce at the minimum the number of stubs and drivers needed to make the system working and we can follow the natural development flow of the components.

The system is composed of different subsystems, so we decided to integrate the components of one subsystem at a time, going to the next one only when the previous one has been completely integrated, starting from the Database Management System, followed by the Business Logic and the External Services, and finally integrating the User and Taxi Client.
The integration order at subsystem level is represented in the following diagram.

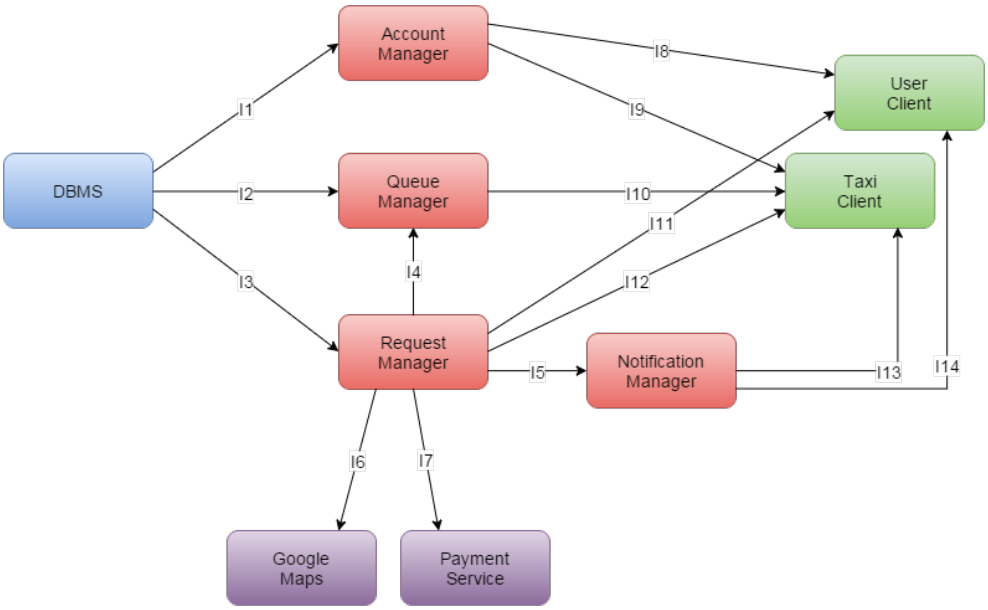


2.4 Sequence of Component Integration

Considering that there is not a proper subsystem integration, we decided to integrate the whole system adding a new component at a time, respecting the strategy given in the previous section.

2.4.1 Software Integration Sequence

Here it's presented the overall integration sequence of all the components, representing the different subsystems with different colours. The arrows are only representative of the integration order of the various components.


Account Manager
User Client
DBMS
Queue Manager
Taxi Client
Request Manager
Notification Manager
Google Maps
Payment Service
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
I12
I13
I14

3 Individual Steps and Test Description

In this section a detailed description of all the test cases is provided, defining Input Specification, Output Specification, Environmental Needs and Tested Interface for each of them.
In the Environmental Needs are reported all the test cases that must already be completed in order to proceed with the current one, and also all the used stubs.
The interfaces reported in the Tested Interface field are defined with all their methods in the Design Document.
The Test Items to be integrated are reported in the form C1 → C2, such that C1 uses an interface exposed by C2.

3.1 Integration test case I1

Test Case Identifier	I1T1
Test Item(s)	Account Manager → DBMS
Input Specification	Test methods defined in the DB interface giving all possible combinations of parameters
Output Specification	The results of the queries are well formed and correct
Environmental Needs	Stubs: S1
Tested Interface	DB

3.2 Integration test case I2

Test Case Identifier	I2T1
Test Item(s)	Queue Manager → DBMS
Input Specification	Test methods defined in the DB interface giving all possible combinations of parameters
Output Specification	The results of the queries are well formed and correct
Environmental Needs	Stubs: S1
Tested Interface	DB

3.3 Integration test case I3

Test Case Identifier	I3T1
Test Item(s)	Request Manager → DBMS
Input Specification	Test methods defined in the DB interface giving all possible combinations of parameters
Output Specification	The results of the queries are well formed and correct
Environmental Needs	Stubs: S1
Tested Interface	DB

3.4 Integration test case I4

Test Case Identifier	I4T1
Test Item(s)	Request Manager → Queue Manager
Input Specification	Test the three methods defined in the TaxiQueue interface, giving all possible input combinations, even the wrong ones
Output Specification	Check that the correct actions are performed or that the right TaxiDriver is returned
Environmental Needs	I2, I3 successfully completed
Tested Interface	TaxiQueues

3.5 Integration test case I5

Test Case Identifier	I5T1
Test Item(s)	Request Manager → Notification Manager
Input Specification	Test the methods defined in the SendNotification interface with all the possible correct and wrong parameter combinations
Output Specification	Check that the notification is sent correctly to the desired client
Environmental Needs	I1-I4 successfully completed
Tested Interface	SendNotification

3.6 Integration test case I6

Test Case Identifier	I6T1
Test Item(s)	Request Manager → Google Maps
Input Specification	Make test calls to GoogleMapsAPI with sample parameters
Output Specification	Check that the returned values are as expected
Environmental Needs	I1-I5 successfully completed
Tested Interface	GoogleMapsAPI

3.7 Integration test case I7

Test Case Identifier	I7T1
Test Item(s)	Request Manager → Payment Service
Input Specification	Make test calls to PaymentAPI with sample parameters
Output Specification	Check that the payment has been correctly completed
Environmental Needs	I1-I5 successfully completed
Tested Interface	PaymentAPI

3.8 Integration test case I8

Test Case Identifier	I8T1
Test Item(s)	User Client → Account Manager
Input Specification	Test the login and creation of accounts operations
Output Specification	All the operations must be performed correctly
Environmental Needs	I1 successfully completed
Tested Interface	Session

3.9 Integration test case I9

Test Case Identifier	I9T1
Test Item(s)	Taxi Client → Account Manager
Input Specification	Test the login and creation of accounts operations
Output Specification	All the operations must be performed correctly
Environmental Needs	I1 successfully completed
Tested Interface	Session

3.10 Integration test case I10

Test Case Identifier	I10T1
Test Item(s)	Taxi Client → Queue Manager
Input Specification	Test confirmation of the availability by the taxi driver
Output Specification	Check the correct insertion of the taxi driver in the queue
Environmental Needs	I1-I6, I9 successfully completed Stubs: S2
Tested Interface	ConfirmAvailability

3.11 Integration test case I11

Test Case Identifier	I11T1
Test Item(s)	User Client → Request Manager
Input Specification	Test ride request, reservation and sharing methods defined in the Request interface
Output Specification	The ride is successfully created
Environmental Needs	I1-I8 successfully completed Stubs: S2
Tested Interface	Request

3.12 Integration test case I12

Test Case Identifier	I12T1
Test Item(s)	Taxi Client → Request Manager
Input Specification	Test the methods defined in the TaxiRequest interface relative to the management of the call
Output Specification	Check that operation is correctly performed
Environmental Needs	I1-I7, I9, I10 successfully completed
Tested Interface	TaxiRequest

3.13 Integration test case I13

Test Case Identifier	I13T1
Test Item(s)	Notification Manager → Taxi Client
Input Specification	Test the notifications calling the methods defined in the NotificationListener interface
Output Specification	Check if the notification has been correctly delivered
Environmental Needs	I1-I7, I9, I10, I12 successfully completed
Tested Interface	NotificationListener

3.14 Integration test case I14

Test Case Identifier	I14T1
Test Item(s)	Notification Manager → User Client
Input Specification	Test the notifications calling the methods defined in the NotificationListener interface
Output Specification	Check if the notification has been correctly delivered
Environmental Needs	I1-I8, I11 successfully completed
Tested Interface	NotificationListener

4 Tools and Test Equipment Required

In this section are presented all the used tools in the different testing phases of the project.

- **JUnit** for unit testing that must be completed before starting the integration test
- **Arquillian** for performing the integration tests defined in this document
- **JMeter** that will be used after the completion of all the integration tests, in order to test the performance of the system with different traffic loads
- **Mosquito** is used to create program stubs used in integration tests

5 Program Stubs and Test Data Required

Here it's presented a list of all the stubs that will be used in the integration tests described in sections 2 and 3.

- [**S1**] **Test database:** database filled with a test set of all the entities defined in the Entity Relationship Schema in the Design Document.
- [**S2**] **GPS locations:** sample GPS locations inside the city.

6 Times spent working on this document

Group Member	**Total Hours**
Andrea Donati	9
Gabriele Carassale	8
Manuel Deleo	8