Integration **T**est **P**lan **D**ocument

Version 1.0

Giorgio Pea(Mat. 853872), Andrea Sessa(Mat. 850082)

21/1/2016

Contents

1 Introduction

1.1 Revision History

Date	Description	Authors
21/01/2016	Delivers of version 1.0	Pea, Sessa

1.2 Purpose

The purpose of the integration test plan is to describe the necessary tests to verify that all the components(see **DD**, reference section) of MyTaxiServive are properly assembled. Integration testing ensures that the unit-tested components interact correctly.

1.3 Scope

The aim of this project is to develop MyTaxiService, a web/mobile application that makes easier and quicker taking taxies within the city's borders. Thanks to MyTaxiService, anyone can request or book a taxi and get realtime information about how long it will take to be picked up or about the taxi's current position and identification code. In addition to that, MyTaxiService provides an efficient way to allocate taxies by dividing the city in zones and using a queue based allocation system, in order to reduce the average waiting time and city's traffic.

1.4 Terms Definition

1.4.1 Glossary

- **Mtaxi:** A taxi that joined MyTaxiService
- **User:** Refers to either a logged registered user or a generic user(see RASD)
- **MyTaxiService(B):** see RASD
- **Administrator:** see RASD
- **Mtaxi bad beahvior**: see RASD

1.4.2 Acronyms

- **DD:** Design Document
- **MVC:** Model View Controller
- **FIFO:** First In First Out

- **API:** Application Programming Interface
- **GUI:** Graphic User Interface
- **GPS:** Global Positioning System
- **CM:** Communication Manager
- **DB:** DataBase

1.5 Reference Documents

- [**RASD**] Requirements and analisys specification document(RASD)
- [**DD**] Design Document(DD)
- [**A1**] Project's assignement number 1
- [**A2**] Project's assignement number 2

2 Integration Strategy

2.1 Overview

In the first part of this section the components to be tested are mentioned, moreover the testing strategy is chosen and the integration testing order defined.
A more detailed specification for each test case is given in the third chapter.

2.2 Entry Criteria

The main entry conditions for this phase is that each of the system components low level functions has been previously subjected to a unit test process.

2.3 Elements to be integrated

For a detailed description of each components function and interaction refer to the **DD**
Three main subsystem can be individuated in the general architecture of MyTaxiService:

1. **Client Side Components**
    - MyTaxiServiceAppUser GUI
    - MyTaxiServiceMYT GUI
    - MyTaxiServiceWebUser GUI
    - MyTaxiServiceWebAdmin GUI
    - MyTaxiServiceAppUser Communication Manager
    - MyTaxiServiceMYT Communication Manager
    - MyTaxiServiceWebUser Communication Manager
    - MyTaxiServiceWebAdmin Communication Manager
2. **Server Side Components**
    - Dispatcher
    - Request Manager
    - External Services Manager
    - Location Manager
    - Queue Manager
    - Request Receiver
3. **Model Side Components**
    - Data Manager

2.4 Integration Testing Strategy

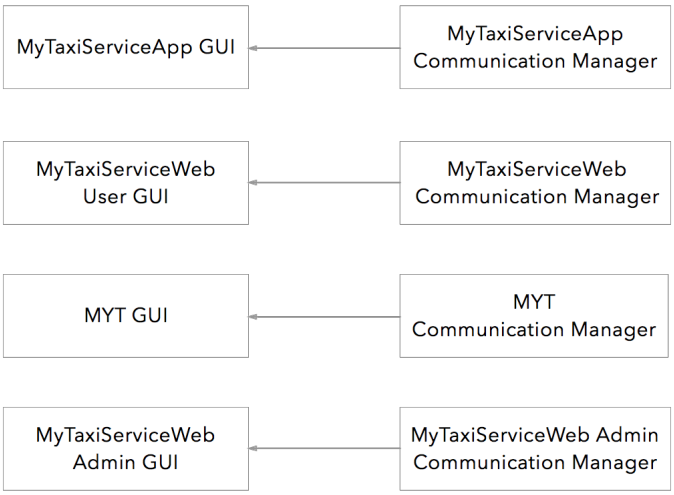
The strategy chosen for the integration of the components of the application is a grouped Bottom-Up integration: for each subsystem S of MyTaxiService, its components are orchestrated together following a bottom-up approach, when this procedure is finished a general integration of the subsystems of the application with each other is performed.
This integration testing strategy has been selected because it mimics very well the structure and the modularity of the software system. The bottom-up approach consists in integrating first those components that are at the low level of the component's dependency tree (see DD's component diagram).

2.5 Sequence of Component/Function Integration

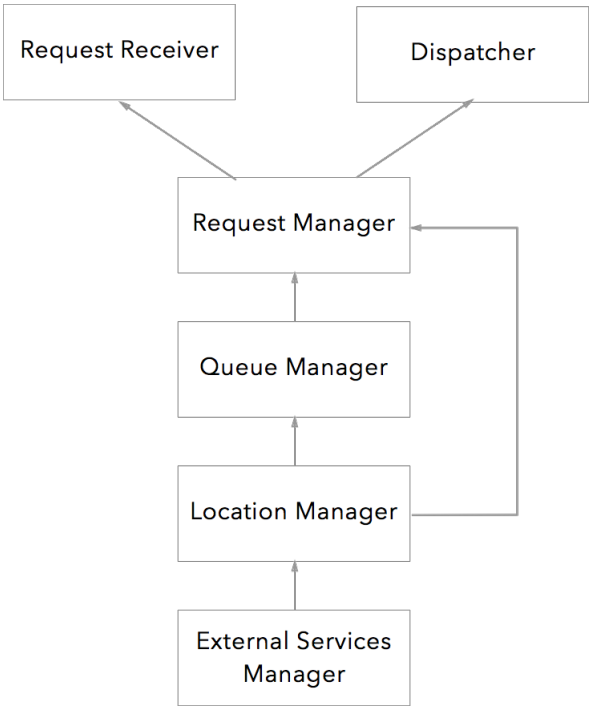
In this section we included the integration sequence for the software components.
The arrows used in the following diagrams reflects the bottom-up integration sequence; in case the arrows does not fix a particular ordering in the integration of some components then the testing phase should give priority to the more critical or more complex component.

2.5.1 Software Integration Sequence

In the following diagram is represented the integration of the View subsystem

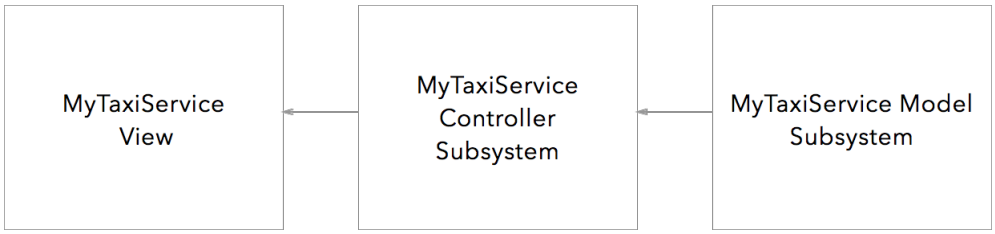


In the following diagram is represented the integration of the Controller subsystem



2.5.2 Subsystem Integration Sequence

In the following diagram is rapresented the general integration sequence between the three subsystems.



3 Individual Step and Test Description

In this section is included the high level description for each step in the integration plan. Each test case correspond with one arrow in the previous diagram(see chapter 2)

3.1 Test Case I1

Test Item(s)	MyTaxiServiceApp CM ⟶ MyTaxiServiceApp GUI
Input Specification	Create the typical input for MyTaxiServiceApp CM
Output Specification	Check if the correct method are called in MyTaxiServiceApp GUI
Environmental Needs	N/A
Purpose	Verifies if MyTaxiServiceApp CM can handle user inputs

3.2 Test Case I2

Test Item(s)	MyTaxiServiceWeb CM ⟶ MyTaxiServiceWeb User GUI
Input Specification	Create the typical input for MyTaxiServiceWeb CM
Output Specification	Check if the correct method are called in MyTaxiServiceWeb User GUI
Environmental Needs	N/A
Purpose	Verifies if MyTaxiServiceWeb CM can handle user inputs coming from the Web site

3.3 Test Case I3

Test Item(s)	MYT CM ⟶ MYT GUI
Input Specification	Create the typical input for MYT CM
Output Specification	Check if the correct method are called in MYT GUI
Environmental Needs	N/A
Purpose	Verifies if MYT CM can handle user inputs coming from a taxi driver

3.4 Test Case I4

Test Item(s)	MyTaxiServiceWeb Admin CM ⟶ MyTaxiServiceWeb Admin GUI
Input Specification	Create the typical input for MyTaxiServiceWeb Admin CM
Output Specification	Check if the correct method are called in MyTaxiServiceWeb Admin GUI
Environmental Needs	N/A
Purpose	Verifies if MyTaxiServiceWeb Admin CM can handle user inputs coming from the admin interface

3.5 Test Case I5

Test Item(s)	External Services Manager ⟶ Location Manager
Input Specification	Create the typical input for External Services Manger
Output Specification	Check if the correct method are called into Location Manager
Environmental Needs	N/A
Purpose	Verifies can correctly provide location data to the Location Manager

3.6 Test Case I6

Test Item(s)	Location Manager ⟶ Queue Manager
Input Specification	Create the typical input for Location Manager
Output Specification	Check if Queue Manger driver receives the correct data according to the input
Environmental Needs	I5 succeeded
Purpose	Verifies if Location Manager produces and formats the location data needed for the Queue Manager

3.7 Test Case I7

Test Item(s)	Location Manager ⟶ Request Manager
Input Specification	Create the typical input for Location Manager
Output Specification	Check if Queue Manger driver receives the correct data according to the input
Environmental Needs	I5 succeeded
Purpose	Verifies if Location Manager produces and formats the location data needed for the Request Manager

3.8 Test Case I8

Test Item(s)	Queue Manager ⟶ Request Receiver
Input Specification	Create the typical input for Queue Manager
Output Specification	Check if Request Receiver driver receives the correct data according to the input
Environmental Needs	I6 and I7 succeeded
Purpose	Verifies if Queue Manager produces and formats the data needed to manage a request

3.9 Test Case I9

Test Item(s)	Request Manager ⟶ Dispatcher
Input Specification	Create the typical input for Queue Manager
Output Specification	Check if correct methods are called in Dispatcher
Environmental Needs	I6 and I7 succeeded
Purpose	Verifies if Request Manager produces the correct answers according to the type of action requested

3.10 Test Case I10

Test Item(s)	Data Manager ⟶ MyTaxiService Controller Subsystem
Input Specification	Create the typical memory situation in the DB and create the typical input for MyTaxiService Controller Subsystem
Output Specification	Check if correct methods are called in Data Manager and the correct data are retrieved
Environmental Needs	All the previous test phase has succeeded
Purpose	Verifies if the Controller Subsystem can retrieve data from the DB system and if the Data Manager can correctly handle data request coming from the Controller

3.11 Test Case I11

Test Item(s)	MyTaxiService Controller Subsystem ⟶ MyTaxiService View Subsystem
Input Specification	Create the typical for MyTaxiService Controller
Output Specification	Check if correct data are retrieved by MyTaxiService View Subsystem
Environmental Needs	All the previous test phase has succeeded
Purpose	Verifies if the View Subsystem can correctly interact with the Controller subsystem

4 Tools and Test Equipment Required

In this section is included a set of suggestion about possible testing tools than could be used to perform the integration testing describe above.
The selection of the tools depends on the selection of the programming language made by the developer.
If the chosen programming language is Java EE:

1. To perform the integration testing between the components the Java EE Arquillian Framework can be used; it provides an environment in which the components under testing are inserted and tested. The specific test protocols will be specified and written in further documents.

2. Manual testing could also be needed to simulate the input of the user in order to generate typical input data used to integrate the various components.

5 Program Stub and Test Data Required

In this section are included any specification of particular input data or component's stub/driver needed to perform the integration steps describe in the previous section.

In order to perform the test case **I10** some anonymized user data should be inserted into the database system.
Appropriates drivers are needed to replace the two external systems: the GPS device and the Traffic Info Services These drivers should simulate external data needed to the External Services Manager to correctly perform the test case **I1**
In general more driver can be introduced: If the integration test phase is run in parallel with the developing phase then some components may no be already available for the integration test phase, such components will be replace with an appropriate driver.

6 Appendix

6.1 Hours Of Work

- Giorgio Pea: xxx hours
- Andrea Sessa: xxx hours

6.2 Tools Used

- Atom/LaTeX: to redact this document