

|  |
| --- |
| Politecnico di Milano |
| Integration Test Plan Document |
| Software Engineering 2: “myTaxyService” |

|  |
| --- |
| 21/01/2016  Stefani Sartini Mat.(853697) |

**Summary**

[1. Introduction 4](#_Toc441134544)

[1. Revision History 4](#_Toc441134545)

[2. Purpose and Scope 4](#_Toc441134546)

[3. List of Definitions and Abbreviations 4](#_Toc441134547)

[4. List of Reference Documents 4](#_Toc441134548)

[2. Integration Strategy 5](#_Toc441134549)

[1. Entry Criteria 5](#_Toc441134550)

[2. Elements to be Integrated 5](#_Toc441134551)

[3. Integration Testing Strategy 5](#_Toc441134552)

[4. Sequence of Component Integration 5](#_Toc441134553)

[1. Software Integration Sequence 6](#_Toc441134554)

[2. Subsystem Integration Sequence 6](#_Toc441134555)

[3. Individual Steps and Test Description 7](#_Toc441134556)

[1. Integration Test Case I1 7](#_Toc441134557)

[2. Integration Test Case I2 7](#_Toc441134558)

[3. Integration Test Case I3 7](#_Toc441134559)

[4. Integration Test Case I4 7](#_Toc441134560)

[5. Integration Test Case I5 7](#_Toc441134561)

[6. Integration Test Case I6 8](#_Toc441134562)

[7. Integration Test Case I7 8](#_Toc441134563)

[8. Integration Test Case I8 8](#_Toc441134564)

[9. Integration Test Case I9 8](#_Toc441134565)

[10. Integration Test Case I10 9](#_Toc441134566)

[11. Integration Test Case I11 9](#_Toc441134567)

[12. Integration Test Case I12 9](#_Toc441134568)

[13. Integration Test Case I13 9](#_Toc441134569)

[14. Integration Test Case I14 10](#_Toc441134570)

[15. Integration Test Procedure TP1 10](#_Toc441134571)

[16. Integration Test Procedure TP2 10](#_Toc441134572)

[17. Integration Test Procedure TP3 11](#_Toc441134573)

[18. Integration test procedure TP4 11](#_Toc441134574)

[4. Tools and Test Equipment Required 11](#_Toc441134575)

[5. Program Stubs, Drivers and Test Data Required 12](#_Toc441134576)

# Introduction

## Revision History

This is the first revision of the document, further may come later in the development of the document or other assignments.

## Purpose and Scope

This document’s purpose is to describe the integration test plan for the myTaxiService application. The focus will be on component interfaces and how they interact, considering that an initial testing of the different code parts’ functionalities has already been done.

The scope of the application is to organize the taxi service of a major city by diving taxi drivers into queues and allowing taxi drivers to request a taxi through the application. The system will notify the nearest driver to the customer with the costumer’s information. Once the driver has accepted the customer is notified with the waiting time and he can accept or decline the incoming taxi. Once the customer has accepted the driver starts moving.

## List of Definitions and Abbreviations

* APPLICATION: The to be developed software “myTaxiService”.
* CLIENT: The part of the application that runs on web browser or on mobile devices. This is the part used by customers and taxi drivers. Contains Registered User, Taxi Driver and Visitor subsystems.
* SERVER: The part of the application that will communicate with the client and handle most of the functionalities. Contains Processing and Data subsystems.
* DATA BASE, DB: The part of the application where all the data is stored and where the taxi queues are stored.
* DATA BASE MANAGEMENT SYSTEM, DBMS: The part of the application that handles all queries on the data base.
* CUSTOMER: Both the taxi drivers and end users: anyone who will use the client application.

## List of Reference Documents

* The Requirement and Specification Document: RASD myTaxiService.Docx
* The Design Document: Design Document.Docx
* The description of the application: Assignment 1 and 2.pdf

# Integration Strategy

## Entry Criteria

The integration phase of the software development is executed after the individual unit testing of the different components of the application. It will focus on how the components connect and communicate with each other, and the interfaces of the application. As such all the single modules and components of the application need to have already been tested with regards to what that particular module will do. This phase of the testing could also be done with white box technique, where the developers of the code write the unit testing for the source code they are writing.

## Elements to be integrated

Chapter 2.3 of the Design document describes the component view of the system, here five main subsystem are defined:

* Processing Sub-System
* Registered User Sub-System
* Taxi Driver Sub-System
* Visitor Sub-System
* Data Sub-System

For each of these subsystem different components were described in the same chapter of the Design Document: The Data subsystem is divided in Data and DBMS, the Processing subsystem is composed of Configuration Interface and Configuration, Processing, Log in and Registration Processing Components. Finally the three client subsystems where divided in User Interface and User manager, Taxi Driver Interface and Taxi Driver Manager, Visitor Interface and Log in and Registration Components. These are all the components that have to be tested for the integration part of the developing of the application. As previously stated all these components need to have already been unit tested, before proceeding with the integration testing.

## Integration Testing Strategy

The chosen strategy for the integration testing is a bottom-up approach. The integration test will start from the interaction of single components of the application, up to the interactions between entire subsystems. This strategy was chosen mainly because most of the components have only one or two interfaces with external components. As such testing the integration of some components will result in having tested the entire subsystem. Furthermore because of the limited number of interfaces, testing the integration and interfaces of the components of the processing subsystem will result in having tested most of the application part. For this same reason, a top-down approach would have similar consequences since testing some subsystems connections would result in testing the single component interfaces.

## Sequence of Component Integration

This chapter will show the order in which the different components and subsystems will be tested. Since the chosen approach is bottom-up, the integration test will start with components and later the subsystems will be tested.

### Software Integration Sequence

The following figure will show the order in which the different components will be integrated within each one of the subsystems.

Data

DBMS

I1

Taxi Driver Interface

Registration

Log in

Visitor Interface

Taxi Driver Manager

User Manager

User Interface

Configuration

Configuration Interface

I2

I3

I4

I6

I5

### Subsystem Integration Sequence

Once the internal components of the different subsystems have been tested, the next phase is to test the integration of the different subsystems. The first two subsystems to be tested will be the Data and Processing Subsystems, in order to verify that the core part of the application functionalities are working. Finally the Processing subsystem with the different Client subsystems will be tested.

Data Subsystem Processing Subsystem Client Subsystem (User/Taxi Driver/ Visitor)

I14

I13

I12

I11

Registration

Log In

Registration Processing

Log In Processing

Taxi Driver Manager

User Manager

I9

I10

I8

I7

DBMS

Configuration

Processing

# Individual Steps and Test Description

In this chapter the integration test cases and procedures will be described.

## Integration Test Case I1

|  |  |
| --- | --- |
| **Test Case Identifier** | I1 |
| **Test Item(s)** | DBMS → Data |
| **Input Specification** | Create typical query input for the DB |
| **Output Specification** | Check if all write and read operations are successful |
| **Environmental Needs** | N/A |

## Integration Test Case I2

|  |  |
| --- | --- |
| **Test Case Identifier** | I2 |
| **Test Item(s)** | Configuration → Configuration Interface |
| **Input Specification** | Create input for the Configuration Interface |
| **Output Specification** | Check that the interface displays the correct information |
| **Environmental Needs** | I1 succeeded |

## Integration Test Case I3

|  |  |
| --- | --- |
| **Test Case Identifier** | I3 |
| **Test Item(s)** | User Manager → User Interface |
| **Input Specification** | Create input for User Interface |
| **Output Specification** | Check that the interface displays the correct information |
| **Environmental Needs** | User Manager Driver |

## Integration Test Case I4

|  |  |
| --- | --- |
| **Test Case Identifier** | I4 |
| **Test Item(s)** | Taxi Driver Manager → Taxi Driver Interface |
| **Input Specification** | Create Input for Taxi Driver Interface |
| **Output Specification** | Check that the interface displays the correct information |
| **Environmental Needs** | Taxi Driver Manager Driver |

## Integration Test Case I5

|  |  |
| --- | --- |
| **Test Case Identifier** | I5 |
| **Test Item(s)** | Registration → Visitor Interface |
| **Input Specification** | Create Input for Visitor Interface |
| **Output Specification** | Check that the interface displays the correct information |
| **Environmental Needs** | Registration Driver |

## Integration Test Case I6

|  |  |
| --- | --- |
| **Test Case Identifier** | I6 |
| **Test Item(s)** | Log In → Visitor Interface |
| **Input Specification** | Create Input for Visitor Interface |
| **Output Specification** | Check that the interface displays the correct information |
| **Environmental Needs** | Log In Driver |

## Integration Test Case I7

|  |  |
| --- | --- |
| **Test Case Identifier** | I7 |
| **Test Item(s)** | Configuration → DBMS |
| **Input Specification** | Create typical queries which Configuration would submit to DBMS |
| **Output Specification** | Check read queries results are correct and accessible by Configuration |
| **Environmental Needs** | I1 and I2 succeeded |

## Integration Test Case I8

|  |  |
| --- | --- |
| **Test Case Identifier** | I8 |
| **Test Item(s)** | Processing → DBMS |
| **Input Specification** | Create typical queries which Processing would submit to DBMS |
| **Output Specification** | Check read queries results are correct, accessible by Processing and check writes on DB were successful |
| **Environmental Needs** | I1 succeeded and Processing Driver |

## Integration Test Case I9

|  |  |
| --- | --- |
| **Test Case Identifier** | I9 |
| **Test Item(s)** | Log In Processing → DBMS |
| **Input Specification** | Create typical queries which Log In Processing would submit to DBMS |
| **Output Specification** | Check read queries results are correct, accessible by Log In Processing and check writes on DB were successful |
| **Environmental Needs** | I1 succeeded and Log In Processing Driver |

## Integration Test Case I10

|  |  |
| --- | --- |
| **Test Case Identifier** | I10 |
| **Test Item(s)** | Registration Processing → DBMS |
| **Input Specification** | Create typical queries which Registration Processing would submit to DBMS |
| **Output Specification** | Check read queries results are correct, accessible by Registration Processing and check writes query on DB were successful |
| **Environmental Needs** | I1 succeeded and Registration Processing Driver |

## Integration Test Case I11

|  |  |
| --- | --- |
| **Test Case Identifier** | I11 |
| **Test Item(s)** | User Manager → Processing |
| **Input Specification** | Create typical input from User Manager to Processing |
| **Output Specification** | Check if correct methods are called in Processing and if correct answers are returned to User Manager |
| **Environmental Needs** | I3 succeeded |

## Integration Test Case I12

|  |  |
| --- | --- |
| **Test Case Identifier** | I12 |
| **Test Item(s)** | Taxi Driver Manager → Processing |
| **Input Specification** | Create typical input from Taxi Driver Manager to Processing |
| **Output Specification** | Check if correct methods are called in Processing and if correct answers are returned to Taxi Driver Manager |
| **Environmental Needs** | I4 succeeded |

## Integration Test Case I13

|  |  |
| --- | --- |
| **Test Case Identifier** | I13 |
| **Test Item(s)** | Registration → Registration Processing |
| **Input Specification** | Create typical input from Registration to Registration Processing |
| **Output Specification** | Check if correct methods are called in Registration Processing and if correct answers are returned to Registration |
| **Environmental Needs** | I5 succeeded |

## Integration Test Case I14

|  |  |
| --- | --- |
| **Test Case Identifier** | I14 |
| **Test Item(s)** | Log In → Log In Processing |
| **Input Specification** | Create typical input from Log In to Log In Processing |
| **Output Specification** | Check if correct methods are called in Log In Processing and if correct answers are returned to Log In |
| **Environmental Needs** | I6 succeeded |

## Integration Test Procedure TP1

|  |  |
| --- | --- |
| **Test Procedure Identifier** | TP1 |
| **Purpose** | This test procedure verifies that the Data Base:   * Can handle input queries * Can output queries results |
| **Procedure Steps** | Execute I1 |

## Integration Test Procedure TP2

|  |  |
| --- | --- |
| **Test Procedure Identifier** | TP2 |
| **Purpose** | This test procedure verifies that the client interface:   * Can handle user input * Can output requested information to the client * Can output user input to processing subsystem * Can receive input from the processing subsystem * Can handle command line input |
| **Procedure Steps** | Execute I2 to I6 |

## Integration Test Procedure TP3

|  |  |
| --- | --- |
| **Test Procedure identifier** | TP3 |
| **Purpose** | This test procedure verifies that all the components from the processing subsystem correctly use the data base. They:   * Correctly output queries to the DB * Can handle input from the DB * Can handle command line input (configuration component) |
| **Procedure Steps** | Execute I7 to I10 |

## Integration test procedure TP4

|  |  |
| --- | --- |
| **Test Procedure Identifier** | TP4 |
| **Purpose** | This test procedure verifies that the processing components:   * Can handle input from client components * Can output information to client components * Can output queries to DBMS component |
| **Procedure Steps** | Execute I11 to I14 |

# Tools and Test Equipment Required

The main tool that will be used for the integration testing is Arquillian. As specified before, all the components that will undergo Integration Testing need to have already been unit tested, with a tool such as Junit. This will help the integration testing step since Arquillian uses similar test cases to Junit. It supports server debugging without needing to deploy the application to the actual server, which will be useful to test the integration of the Data Subsystem with the Processing subsystem and later the Processing Subsystem with the Client Subsystems.

The first part of the integration test (I2 to I5) will be between the interface components and their manager components, this part should be manually tested. This was chosen because the client subsystems are mainly user interfaces and manually testing these user interfaces guarantees not only that the correct functions or methods are called, but also that the application displays the correct data. This allows whoever is in charge of the testing to immediately identify problems in the subsystems through visual feedback. Testing will simply be running the client part of the application and verifying that the displayed information is the consistent to what the processing subsystem received or sent to the client.

# Program Stubs, Drivers and Test Data Required

No stubs are used during the integration test of this application. On the other hand many drivers will be used to emulate functions or components that will require the functionalities of other components. Examples of this are I2 to I5 in which the user interfaces communication with their manager components will be tested. Since the integration between the manager components and the processing subsystem is still not tested, drivers will be needed to simulate what and how the processing subsystem will be communicating with the different Manager Components. In the same way, once the integration between the Processing subsystem and the Data subsystem takes place, (I7 to I10) the drivers of the processing components are needed to give information to the components on what to do. These drivers will simulate the requests the subsystem receives from the client subsystem. Finally in the cases form I11 to I14 no drivers are needed since the integration of manager components and user interfaces has already taken place, so the user interface components can be used to communicate with the user manager components. For I1, no driver is needed since the verification of the correct behaviour of the communication of the two involved components can be tested running a series of queries on the DB and verifying the results.

Some data must be given to the different components before the test can take place: for example the different zones in which the city is divided must be saved in the DB. For testing purpose, these zone could be fake or real zones of the city. In this way when testing the integration between the DB and the Processing component, this last will be able to as the DB for zones and determine in which the taxi drivers should be placed. At this point the test will be successful if the drivers are added to the correct zone’s queue. Another example of test data that is required for this testing phase is a list of accounts. In this way, when testing the interaction with the DB, the Log In or Register Processing components will be able to determinate if an account already exists or if it can be created.