A.Y. 2015/2016

POLITECNICO DI MILANO

COMPUTER SCIENCE AND ENGINEERING

SOFTWARE ENGINEERING 2

MyTaxiService

Integration Test Plan Document

Authors

SARA PISANI – 854223

LEONARDO TURCHI – 853738

21/01/2016

MyTaxiService-TestPlan.pdf · V 0.5

Table of contents

AGGIORNARE SOMMARIO

1. Introduction

1.1 Revision History

This document is at:

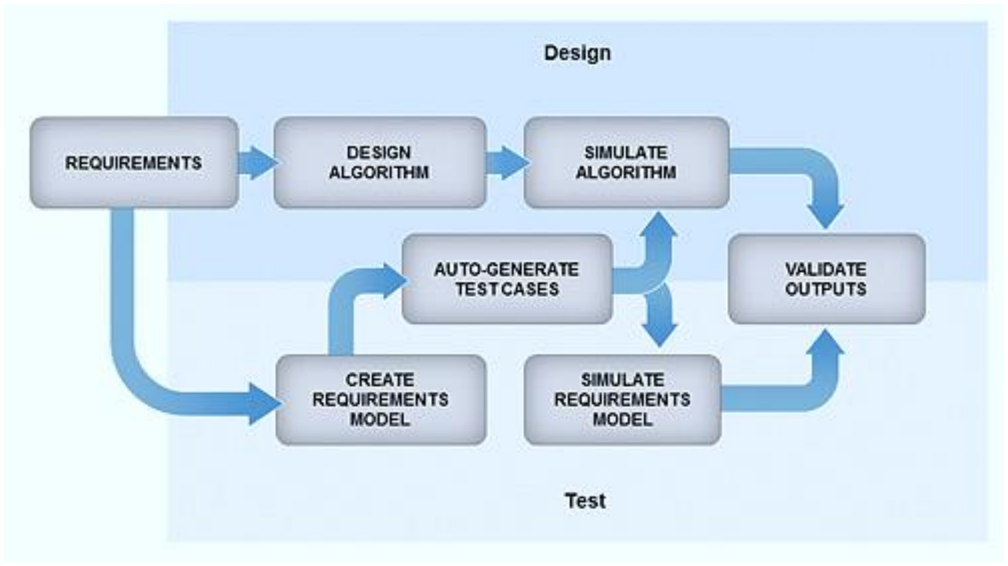
Version: 1

Revision: 1

Status: draft (academic document – waiting for approval)

1.2 Purpose and Scope

This document aims to describe the plans for testing the integration of the created components.
The purpose is to test the component's interfaces explained in the RASD and DD attached documents. We start by giving an overview of the entire system, with an highlight of the subsystems to be integrated.



All the application/algorithm/data interfaces will be tested in accordance with the steps described in this document.
In the following, we are going to explain (to the development team) what to test and which tools are needed for testing.
This document is for the development team, to give them the directives of the stuff to be tested.

1.3 List of Definitions and Abbreviations

- *RASD – Requirements Analysis and Specification Document*
- *DD – Design Document*
- *ID – Inspection Document*
- *ITPD – Integration Test Plan Document*

- *API – Application Program Interface*
- *JVM – Java Virtual Machine*
- *J2EE – Java 2 Enterprise Edition*
- *DBMS – DataBase Management System*
- *OS – Operative System*

- *UML – Unified Modeling Language*
- *DEV – Development*
- *IEEE – Institute of Electrical and Electronics Engineers*

1.4 List of Reference Documents

List of all references:

- [*pdf*] Assignment rules and group registration
- [*pdf*] Assignment 4 - Test Plan
- [*pdf*] RASD Document
- [*pdf*] DD Document

- [*pdf*] Inspection Document

2. Integration Strategy

2.1 Entry Criteria

The criteria that must be met, before integration testing of specific elements may begin, are:

- *RASD* and *DD* documents are complete and revised.

- Most of *MyTaxiService* modules must be complete.
  Is not necessary that all code is complete because is possible to test adding code integration (Stub or Drivers, depending on the chosen strategy), but a substantial percentage of components are required to start the testing phase.

- Driver to cover the eventual lack of modules.

- The implementation must satisfy the requirements and the assumptions specified in the RASD document.

- The implementation must be done following the architecture and the design specified in the DD document.

2.2 Elements to be Integrated

The elements that have to be integrated are the ones specified in the Component View in chapter 2.3 of the *DD*.

This document explain how they have to be integrated and the order of integration, according to the strategy adopted.

ALTRO??

2.3 Integration Testing Strategy

In this case a Bottom-Up strategy should be better: in this approach testing is conducted from sub module to main module, if the main module is not developed a temporary program called Drivers is used to simulate the main module.

Advantages are:

- ✓ Advantageous if major flaws occur toward the bottom of the program.
- ✓ Test conditions are easier to create (develop drivers is simpler than develop stubs).
- ✓ Observation of test results is easier.
- ✓ Bugs are more easily found.
- ✓ Makes it easier to report testing progress in the form of a percentage.
- ✓ Helps to determine the levels of software developed.

Even if driver components have to be created starting from zero, is wrong to consider them as throwaway code, because Drivers can become automated test cases.

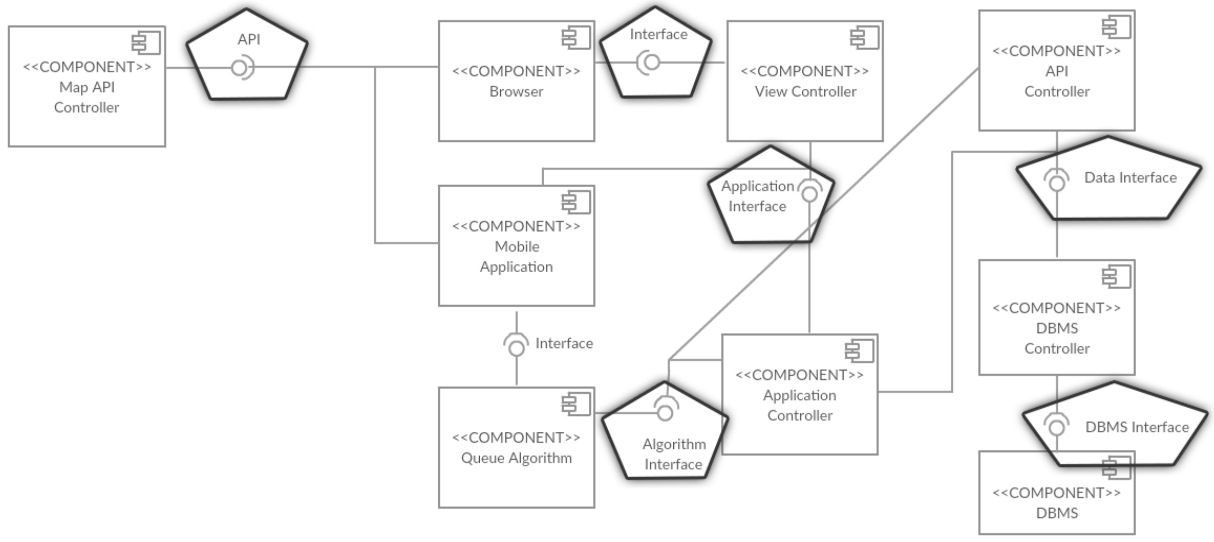
2.4 Sequence of Component/Function Integration

In this section we will identify all the integration sequences of the various modules.
An arrow (A → B) will be used to explain the integration that goes from a component A to a component B, while A *USES* the interface of the component B.

2.4.1 Software Integration Sequence

Note: the system is unique and does not have subsystems; this chapter will be focused on the main system integration.

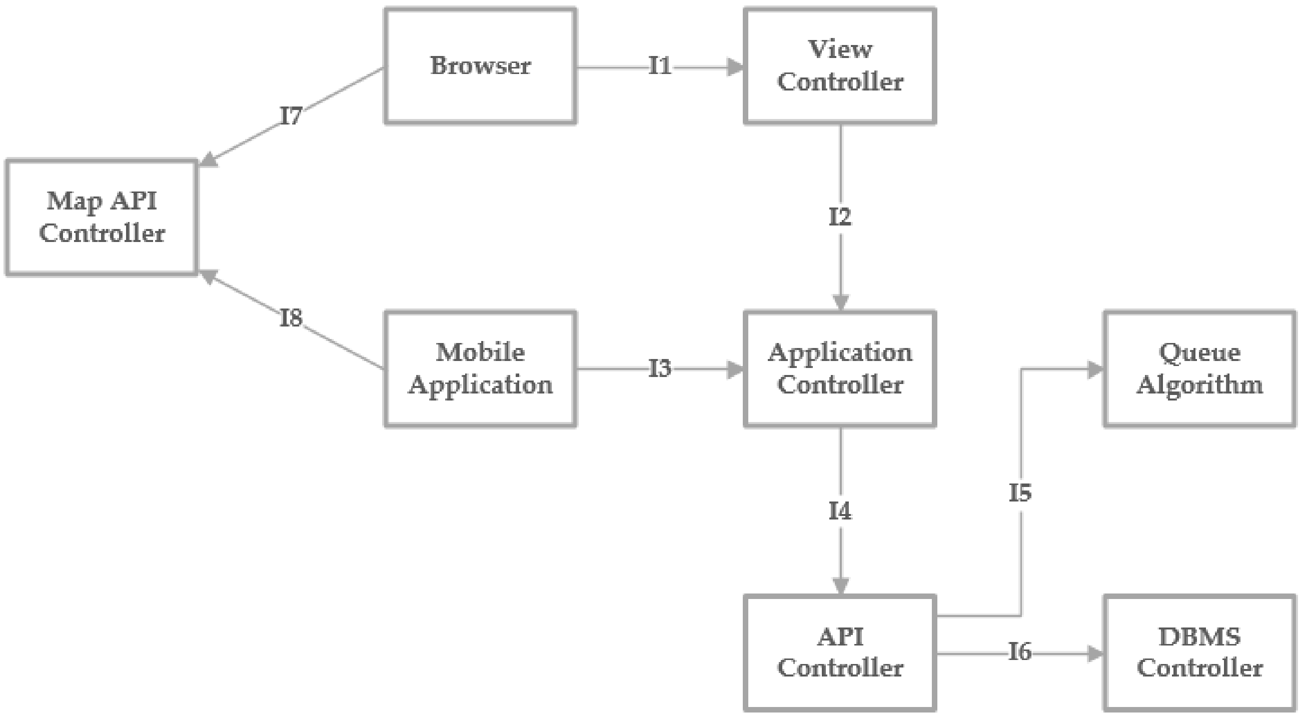
In line with the Design Document, we have the following highlighted interface to test:


<<COMPONENT>> Map API Controller
API
<<COMPONENT>> Browser
Interface
<<COMPONENT>> View Controller
<<COMPONENT>> API Controller
Application Interface
Data Interface
<<COMPONENT>> Mobile Application
Interface
<<COMPONENT>> DBMS Controller
<<COMPONENT>> Application Controller
<<COMPONENT>> Queue Algorithm
Algorithm Interface
DBMS Interface
<<COMPONENT>> DBMS

These interfaces make all the components working through a data communication.

The picture below shows the interaction and the communication between the components, and the integration testing that will be done.

In this case, we recognize 8 test cases, as you can see in the following picture.



ID	Integration Test	Ref
I1	Browser → View Controller	*3.1.1*
I2	View Controller → Application Controller	*3.1.2*
I3	Mobile Application → Application Controller	*3.1.3*
I4	Application Controller → API Controller	*3.1.4*
I5	API Controller → Queue Algorithm	*3.1.5*

I6	API Controller → DBMS Controller	*3.1.6*
I7	Browser → Map API Controller	*3.1.7*
I8	Mobile Application → Map API Controller	*3.1.8*

3. Individual Steps and Test Description

3.1 Integration Test Cases

3.1.1 Integration Test – Case I1

Test case identifier	**I1**
Test items	Browser → View Controller
Input specification	login simulation, authentication data, page navigation, input buttons, page details view, 'back' functionality (all the requests coming from the browser) [RASD 3.5.3]
Output specification	Correct response for each type of request. No browser's visualization issue. No connection error (GET requests check).
Environmental needs	I2 succeeded.

3.1.2 Integration Test – Case I2

Test case identifier	**I2**
Test items	View Controller → Application Controller
Input specification	request for a page, request for a response, execution request for a login, execution request for a 'new ride' page's data, correct popup visualization dispatch (all the requests coming from the view)
Output specification	Correct response for each type of request.

	No error or deadlock.
Environmental needs	I4 succeeds. The view controller must be available and in correct communication with the application controller to dispatch the requests. This test case is similar to the I3.

3.1.3 Integration Test – Case I3

Test case identifier	**I3**
Test items	Mobile Application → Application Controller
Input specification	login simulation, authentication, various app section navigation, response for an input (all the requests coming from the app) [RASD 3.5.3]
Output specification	Correct response for each type of request. No communication error (http request check).
Environmental needs	I4 succeeds. The application controller must be reachable and in correct communication with the DBMS controller to dispatch all the requests. Database data available (even simulated data). This test case is similar to the I2.

3.1.4 Integration Test – Case I4

Test case identifier	**I4**
Test items	Application Controller → API Controller
Input specification	method call for data get or queue management, forwarding query, responding with data, stress test with many simultaneous requests
Output specification	Correct forwarding of requests, no error thrown. No communication error (GET/POST request check).
Environmental needs	The API controller must be available.

3.1.5 Integration Test – Case I5

Test case identifier	**I5**
Test items	API Controller → Queue Algorithm
Input specification	request for a queue management, execution of a enqueue/dequeue/migration method [DD 3.2 – 3.3]
Output specification	Correct execution of commands. Real change of the queue, without exception or overflows.
Environmental needs	I4 succeeds. The API controller must be ready. Queue data structure must exist, must be available and ready (even with simulated data).

3.1.6 Integration Test – Case I6

Test case identifier	**I6**
Test items	API Controller → DBMS Controller
Input specification	database authentication, request for data, *INSERT* - *SELECT* - *UPDATE* commands (all commands for a correct DB management)
Output specification	Correct commands execution, no exception thrown. No connection error to DBMS.
Environmental needs	I4 succeeds. The API controller must be ready. DBMS controller must be available.

	The DBMS core (data structure) must be available and ready to serve data (even simulated data).

3.1.7 Integration Test – Case I7

Test case identifier	**I7**
Test items	Browser → Map API Controller
Input specification	get position from coordinates, get street/zone from coordinates, get map thumbnail from coordinates, calculate approximate trip time, check correctness of an address, check if address is inside an area [DD 4.1 – 4.2]
Output specification	Correct dispatch for all requests, no error thrown. API-Key for the public service (i.e. G.Maps) are valid for many simultaneous requests. No connection errors.
Environmental needs	I1, I2 succeeds. Public Map-API service is available and reachable. This test case is similar to the I8.

3.1.8 Integration Test – Case I8

Test case identifier	**I8**
Test items	Mobile Application → Map API Controller
Input specification	get position from coordinates, get street/zone from coordinates, get map thumbnail from coordinates, calculate approximate trip time, check correctness of an address, check if address is inside an area

	[DD 4.1 – 4.2]
Output specification	Correct dispatch for all requests, no error thrown. API-Key for the public service (i.e. G.Maps) are valid for many simultaneous requests. No connection errors.
Environmental needs	I3 succeeds. Public Map-API service is available and reachable. This test case is similar to the I7.

3.2 Test Procedures

3.2.1 Integration Test – Procedure TP1

Test procedure identifier	**TP1**
Purpose	User Registration and User Login … [RASD 3.5 – 3.5.3] [DD 2.5]
Procedure steps	

3.2.2 Integration Test – Procedure TP2

Test procedure identifier	**TP2**
Purpose	Ride Request and Ride Booking … [RASD 3.5 – 3.5.3] [DD 2.5]
Procedure steps	

3.2.3 Integration Test – Procedure TP3

Test procedure identifier	**TP3**

Purpose	Driver Area Management … [RASD 3.5 – 3.5.3] [DD 2.5]
Procedure steps	

4. Tools and Test Equipment Required

Identify all tools and test equipment needed to accomplish the integration. Refer to the tools presented during the lectures. Explain why and how you are going to use them. Note that you may also use manual testing for some part. Consider manual testing as one of the possible tools you have available.

4.1 Aaaaaaa

---->

((IN BASE AI TEST SOPRA))

4.2 Software and tools

- Microsoft Word: to redact this document.
- GitHub: to share the material of this project.
- VirtualBox: to open virtual machine.
- Eclipse Luna: to open Glassfish's code.
- Java/Glassfish Online Documentation.

5. Program Stubs and Test Data Required

Based on the testing strategy and test design, identify any program stubs or special test data required for each integration step.

Integration Test Plan Example
https://beep.metid.polimi.it/documents/3343933/5b3768d0-d949-4369-87e1-7a31b6943726

6. Hours of works

Here is the time spent for redact this document:

[sum of hours spent by team's members]

+2h (14/01)

+6h (15/01)

+2h (18/01)

+6h (19/01)

+6h (20/01)

+2h (21/01)

TOTAL ~ 40 hours

- Leonardo Turchi: ~ 20 hours
- Sara Pisani: ~ 20 hours