Software Engineering 2

Test Plan Document

Authors
Paolo Paterna, Lara Premi

Politecnico di Milano, Italy
Email: paolo.paterna@mail.polimi.it, lara.premi@mail.polimi.it

21th Jan 2016

Contents

1 Introduction

1.1 Purpose and Scope

1.1.1 Purpose

The purpose of this document is describing how you plan to accomplish the integration test. All the components of the system have to be tested accordingly to the appropriate method described in this document.

1.1.2 Scope

The scope of this system is to manage taxis in a city. A town is divided in zone of 2 square kilometers and, for each zone, the system defines a queue, composed by the identifier, which is the vehicle plate, of free taxis in that specific zone. A user can require a taxi ride from a zone, but can also book one for another moment, using the web application or the mobile one. About long-term reservations, the user can also, after have created one, modify the date or the hour or both of his/her booking, and he/she can delete it.

1.2 Reference Documents

- *Assignment 4 - integration test plan.pdf* file on BeeP university platform.
- *Integration plan test example.pdf* file on BeeP university platform.
- *08-verification second lecture.pdf* file on BeeP university platform.
- Design Document from a previous delivery.

2 Integration Strategy

2.1 Entry Criteria

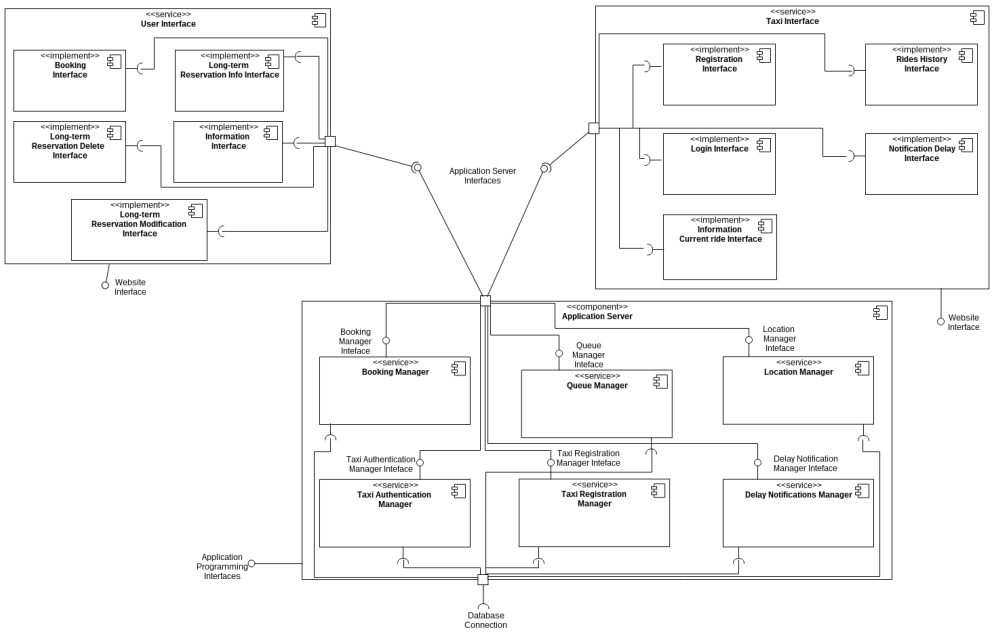
Before starting the test phase, the following documents must be ready and read:

- The Design Document of previous delivery to understand how the software is built, in particular, the component view which specifies how the various modules of the system are connected to each other.
- Sections 2.3 and 2.4 of this document that explains which test strategy we use and the testing order of every module and submodule.
- How the database is built and which data it contains.

  Also before testing each module that comes after some other modules in the testing order:

- Test result of previous order and the drivers required to run that test.

2.2 Elements to be Integrated



The system to be tested is represented in this image. Every submodule has to be tested after all his predecessors in testing order, specified in section

2.5, have been tested. The system can be divided in 3 modules, and every module can be divided in submodules.

2.2.1 Modules to be Tested

- **Application Server**
  - Booking Manager
  - Taxi Authentication Manager
  - Queue Manager
  - Taxi Registration Manager
  - Location Manager
  - Delay Notification Manager
- **Taxi Interface**
  - Registration Interface
  - Login Interface
  - Information Current Ride Interface
  - Rides History Interface
  - Notification Delay Interface
- **User Interface**
  - Booking Interface
  - Long-term Reservation Delete Interface
  - Long-term Reservation Info Interface
  - Information Interface
  - Long-term Reservation Modification Interface

2.3 Integration Testing Strategy

The strategies we could use are: top-down, bottom-up, sandwich, thread and critical modules. After some considerations, we have understood that, among them, the preferable strategies are the first two ones written. Our choice depends on the three facts below:

- besides the sandwich strategy is flexible and adaptable, it's complicated to plan;

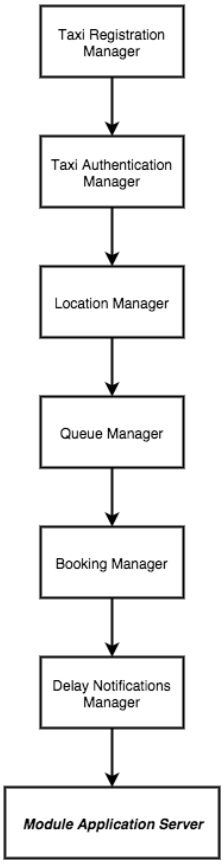
- it's not better to use the thread strategy: if we consider portions of several modules, some problems can occur because some modules depend on others, entirely and not only on a part of them.
- the critical modules strategy is more imprecised than the others.

It's indifferent to use the bottom-up strategy or the top-down one. In this document, we have focused on the first strategy nominated.

2.4 Sequence of Component/Function Integration

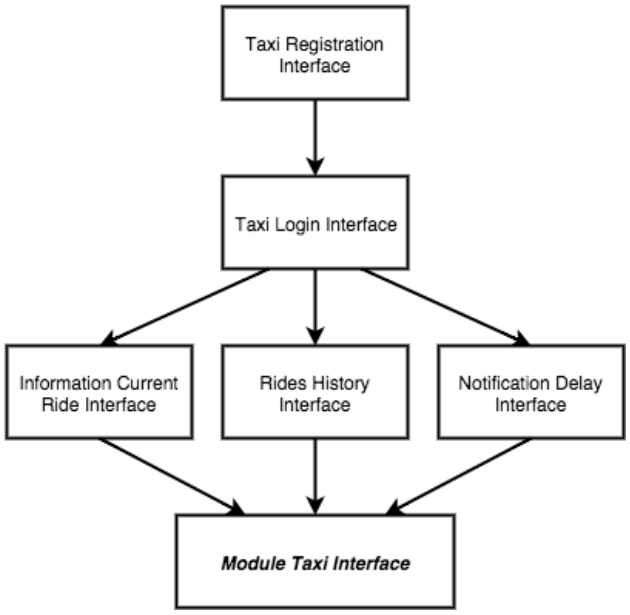
2.4.1 Integration Sequence

Considering the bottom-up strategy, first of all, we will test the module called "Application Server", and, in particular, its submodules, represented in the image below:

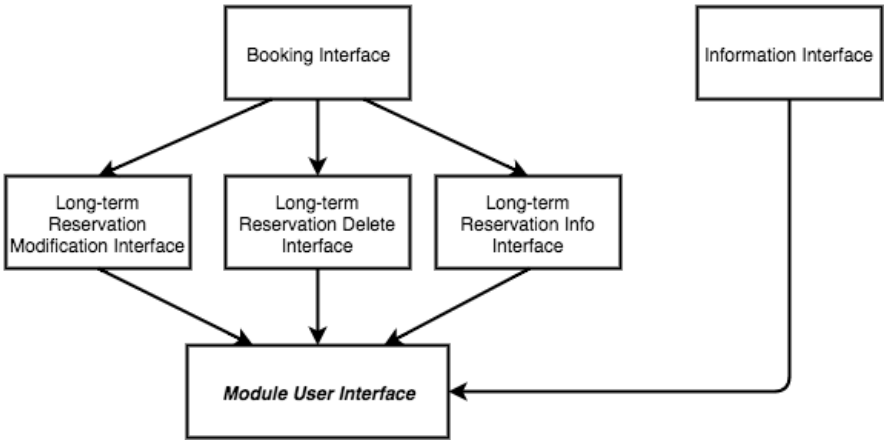


After that, we can test two modules:

- the one named "Taxi Interface", represented in the image below



- the one called "User Interface", corrisponding to the diagram below



The arrows represent the testing order.

3 Individual Steps and Test Description

Remember that the white-box testing and the black-box one could be done because the first is suitable for Unit testing and the second is suitable for integration, allowing also to identify if there are some bugs (white box) or missing/misbehaving functionalities in the system.

3.1 Integration test case I1

Test Case Identifier	I1T1
Test Item(s)	Taxi Registration Manager → Taxi Authentication Manager
Input Specification	Create typical Taxi Registration Manager input
Output Specification	Check if the correct functions work well in Taxi Authentication Manager
Environmental needs	Unit test completed successfully and the Location Manager Driver

3.2 Integration test case I2

Test Case Identifier	I2T1
Test Item(s)	Taxi Authentication Manager → Location Manager
Input Specification	Create typical Taxi Authentication Manager input
Output Specification	Check if the correct functions work well in Location Manager
Environmental needs	I1 succeeded

3.3 Integration test case I3

Test Case Identifier	I3T1
Test Item(s)	Location Manager → Queue Manager
Input Specification	Create typical Location Manager input
Output Specification	Check if the correct functions work well in Queue Manager
Environmental needs	I2 succeeded

3.4 Integration test case I4

Test Case Identifier	I4T1
Test Item(s)	Queue Manager → Booking Manager
Input Specification	Create typical Queue Manager input
Output Specification	Check if the correct functions work well in Booking Manager
Environmental needs	I3 succeeded

3.5 Integration test case I5

Test Case Identifier	I5T1
Test Item(s)	Booking Manager → Delay Notifications Manager
Input Specification	Create typical Booking Manager input
Output Specification	Check if the correct functions work well in Delay Notifications Manager
Environmental needs	I4 succeeded

3.6 Integration test case I6

Test Case Identifier	I6T2
Test Item(s)	Taxi Registration Interface → Taxi Login Interface
Input Specification	Create typical Taxi Registration Interface input
Output Specification	Check if the correct functions work well in Taxi Login Interface
Environmental needs	Application Server

3.7 Integration Test Case I7

Test Case Identifier	I7T2
Test Item(s)	Information Current Ride Interface → Module Taxi Interface
Input Specification	Create typical Information Current Ride Interface Input
Output Specification	Check if the correct methods work well in the Module Taxi Interface
Environmental Needs	I6 succeeded

3.8 Integration Test Case I8

Test Case Identifier	I8T2
Test Item(s)	Ride History Interface → Module Taxi Interface
Input Specification	Create typical Ride History Interface Input
Output Specification	Check if the correct methods work well in the Module Taxi Interface
Environmental Needs	I6 succeeded

3.9 Integration Test Case I9

Test Case Identifier	I9T2
Test Item(s)	Notification Delay Interface → Module Taxi Interface
Input Specification	Create typical Notification Delay Interface Input
Output Specification	Check if the correct methods work well in the Module Taxi Interface
Environmental Needs	I6 succeeded

3.10 Integration Test Case I10

Test Case Identifier	I10T3
Test Item(s)	Booking Interface → Long-Term Reservation Modification Interface
Input Specification	Create typical Booking Interface Input
Output Specification	Check if the correct methods work well in the Long-Term Reservation Modification Interface
Environmental Needs	I5 succeeded

3.11 Integration Test Case I11

Test Case Identifier	I11T3
Test Item(s)	Booking Interface → Long-Term Reservation Delete Interface
Input Specification	Create typical Booking Interface Interface Input
Output Specification	Check if the correct methods work well in the Long-Term Reservation Delete Interface
Environmental Needs	I5 succeeded

3.12 Integration Test Case I12

Test Case Identifier	I12T3
Test Item(s)	Booking Interface → Long-Term Reservation Info Interface
Input Specification	Create typical Booking Interface Interface Input
Output Specification	Check if the correct methods work well in the Long-Term Reservation Info Interface
Environmental Needs	I5 succeeded

3.13 Integration Test Case I13

Test Case Identifier	I13T3
Test Item(s)	Long-Term Reservation Modification Interface → Module User Interface
Input Specification	Create typical Long-Term Reservation Modification Interface Input
Output Specification	Check if the correct methods work well in the Module User Interface
Environmental Needs	I10 succeeded

3.14 Integration Test Case I14

Test Case Identifier	I14T3
Test Item(s)	Long-Term Reservation Delete Interface → Module User Interface
Input Specification	Create typical Long-Term Reservation Delete Interface Input
Output Specification	Check if the correct methods are called in the Module User Interface
Environmental Needs	I11 succeeded

3.15 Integration Test Case I15

Test Case Identifier	I15T3
Test Item(s)	Long-Term Reservation Info Interface → Module User Interface
Input Specification	Create typical Long-Term Reservation Info Interface Input
Output Specification	Check if the correct methods are called in the Module User Interface
Environmental Needs	I12 succeeded

3.16 Integration Test Case I16

Test Case Identifier	I16T3
Test Item(s)	Information Interface → Module User Interface
Input Specification	Create typical Information Interface Input
Output Specification	Check if the correct methods are called in the Module User Interface
Environmental Needs	I5 succeeded

4 Tools and Test Equipment Required

To test our system we can use a tool like *JUnit* to verify the behavior of the system submodules. This method should help to reduce the number of bugs in every module.

To verify the behavior of every module we can use a tool like *mockito*; in this way we can test the integration of every submodule.

Once the system is fully tested using the tools above, the performance of the system can be analyzed using a software like *Apache JMeter*; in this way an average loads or a stress test can be run to see how the system behaves in every situation.

5 Program Stub and Test Data Required

We have considered some stubs and drivers for our system.

5.1 Module Application Server

According to this module, the stubs are represented in the list below:

- one for the "Taxi Authentication Manager", that simulates the "Taxi Registration Manager";
- one for the "Location Manager", that simulates the "Taxi Authentication Manager";
- one for the "Queue Manager", that simulates the "Location Manager";
- one for the "Booking Manager", that simulates the "Queue Manager";
- one for the "Delay Notifications Manager", that simulates the "Booking Manager";
- one for the whole module, that simulates the "Delay Notifications Manager".

Instead, the drivers are:

- one for the "Taxi Registration Manager";
- one for the "Taxi Authentication Manager";
- one for the "Location Manager";

- one for the "Queue Manager";
- one for the "Booking Manager";
- one for the "Delay Notifications Manager";
- one for the whole module.

5.2 Module Taxi Interface

According to this module, the stubs are represented in the list below:

- one for the "Taxi Login Interface", that simulates the "Taxi Registration Interface";
- one for the "Information Current Ride Interface", that simulates the "Taxi Login Interface";
- one for the "Rides History Interface", that simulates the "Taxi Login Interface";
- one for the "Notification Delay Interface", that simulates the "Taxi Login Interface";
- one for the whole module, that simulates the "Information Current Ride Interface", "Rides History Interface" and "Notification delay Interface".

Instead, the drivers are:

- one for the "Taxi Registration Interface";
- one for the "Taxi Login Interface";
- one for the "Information Current Ride Interface";
- one for the "Rides History Interface";
- one for the "Notification Delay Interface";
- one for the whole module.

5.3 Module User Interface

According to this module, the stubs are represented in the list below:

- one for the "Long-term Reservation Modification Interface", that simulates the "Booking Interface";
- one for the "Long-term Reservation Delete Interface", that simulates the "Booking Interface";
- one for the "Long-term Reservation Info Interface", that simulates the "Booking Interface";
- one for the whole module, that simulates the "Long-term Reservation Modification Interface", the "Long-term Reservation Delete Interface", the "Long-term Reservation Info Interface" and the "Information Interface"

Instead, the drivers are:

- one for the "Booking Interface";
- one for the "Long-term Reservation Modification Interface";
- one for the "Long-term Reservation Delete Interface";
- one for the "Long-term Reservation Info Interface";
- one for the "Information Interface";
- one for the whole module.

6 Hours of Work

In order to write this document, we have done the following hours of work:

- Paolo Paterna: 10 Hrs;
- Lara Premi: 10 Hrs.