COMPUTER SCIENCE AND ENGINEERING

PROJECT OF SOFTWARE ENGINEERING 2

Integration Test Plan

POLITECNICO
MILANO 1863

Author: Riccardi Vincenzo 793241

Reference Professor: Mirandola Raffaela

Summary

1.Introduction

Purpose

The Test Plan Document (ITPD) describes how I plan to accomplish the integration test.

The purpose of this document is to test the interaction between all the components of application.

Scope

The principal scope is to create an application with as few bugs as possible. On the other hand, the application must be easy to use, usability is one of the most important characteristic of it.

List of definition and abbreviations

- RASD: Requirements Analysis and Specification Document.
- DD: Design Document.
- MVC: Model-View-Controller Pattern.
- User: a person who has yet done registration and who can use all the functionalities.
- Visitor: a person who visits the application without being logged in.
- Application: term used to indicate software system which will be tested.

List of reference documents

- Requirements Analysis and Specification Document.
- Design Document.
- Code Inspection Document.
- Assignment 1,2,3,4.
- IEEE Recommended Practice for Software Requirements Specifications.

All the files can be found on beep portal:

https://beep.metid.polimi.it

2.Integration Strategy

Entry Criteria

Some test sets are identified by a test criterion.

A test set satisfies a criterion if it is in the domain of criterion. To identify test sets I will use the black-box method, so test sets are based on the knowledge of some specifics of the system.

The most important criterion that I will adopt is “transition coverage” (one test for each transition between states).

Also a transition can be thought of as pair (precondition, postcondition).

Element to be integrated

To identify the various transition between the states it refers to “State Machine Diagram” contains in chapter 5 of RASD.

The principal transitions are:

- The client request to log into the application.
- The client request to register in the application.
- The client request to call a taxi.
- The client request to have an estimation of journey.

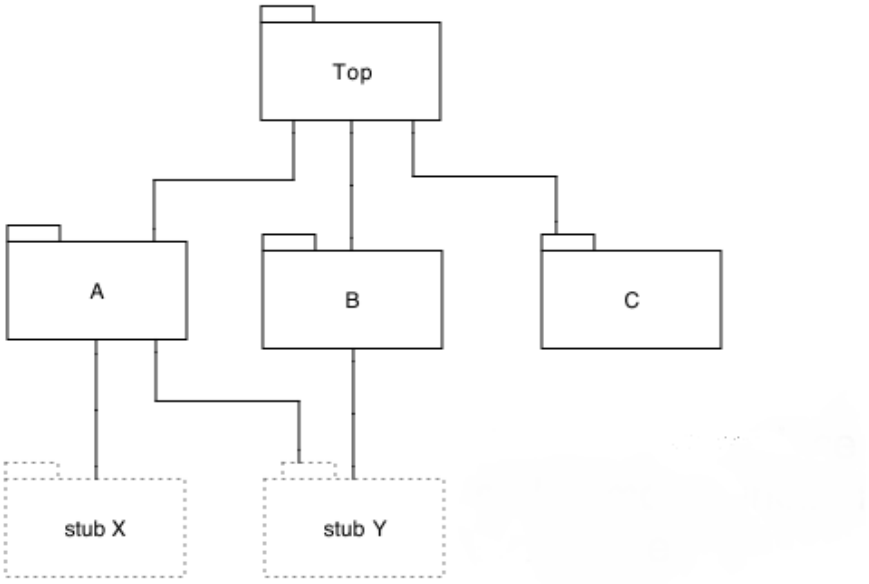
Behind this, there is of course the structure on the MVC model, with all communications between client and server.

Integration Testing Strategy

I use the structural strategy to integration testing, I chose in particular the top-down strategy.

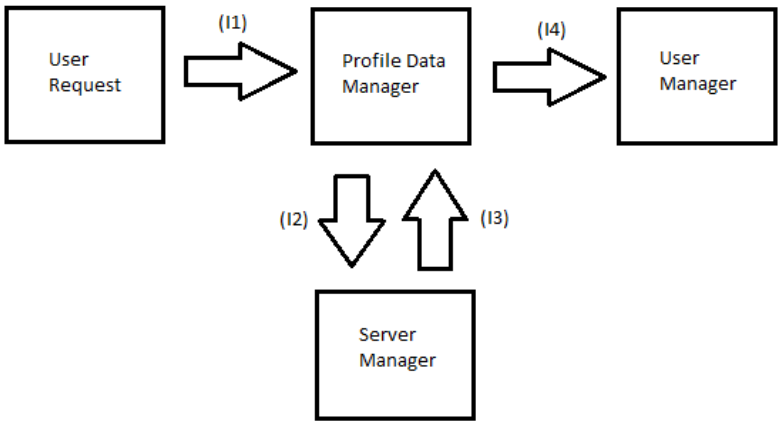
The structural strategy is based on a hierarchical project structure and all modules are constructed, integrated and tested following it.

I preferred the structural strategies because are simpler than functional strategies, indeed the thread and critical modules are preferred for larger subsystems. No driver required if program tested from top-level to bottom, through the use of the stub all functionality can be tested.



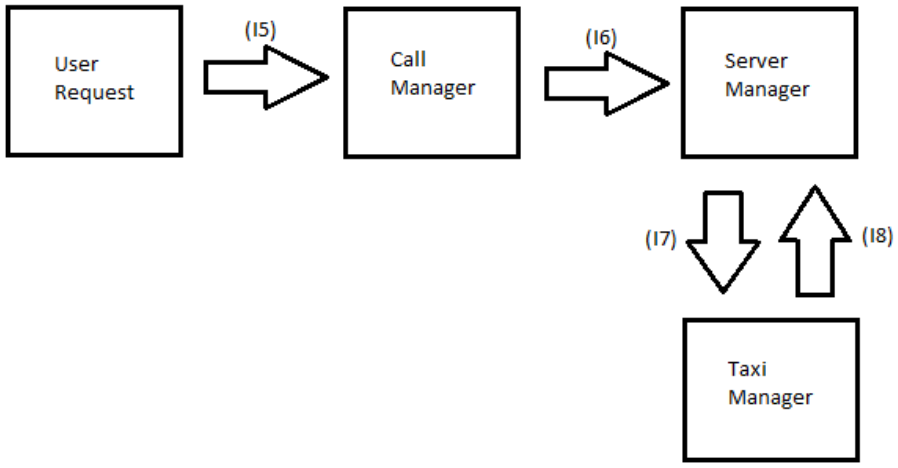
Software Integration Sequence

Integration test of Log In and Registration functionalities:



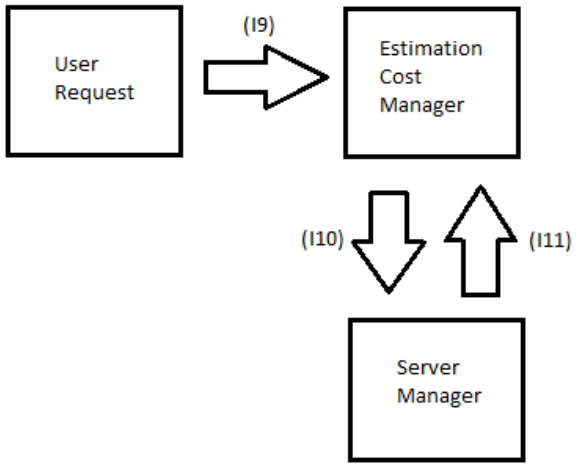
Id	Integration test
I1	User communicates with Profile Data Manager
I2	Profile Data Manager sends request to the Server
I3	Server sends answer to the Profile Data Manager
I4	Profile Data Manager communicates with User Manager

Integration test of call taxi functionality:



Id	Integration test
I5	User communicates with Call Manager
I6	Call Manager sends request to the Server
I7	Server Communicates with Taxi Manager
I8	Taxi Manager answers to the Server

Integration test of Estimation cost functionality:



Id	Integration test
I9	User communicates with Estimation Cost Manager
I10	Estimation Cost Manager sends request to the Server
I11	Server Communicates with Estimation Cost Manager

3.Individual Steps and Test Description

Integration test case I1

Test Case Identifier	I1T1
Test Item(s)	User → Profile Data Manager
Input Specification	Create a User Request
Output Specification	Check if the correct functions are called in Profile Data Manager
Environmental Needs	User Driver

Integration test case I2

Test Case Identifier	I2T1
Test Item(s)	Profile Data Manager → Server Manager
Input Specification	Send User Request
Output Specification	Check if the correct functions are called in Server Manager
Environmental Needs	I1 succeeded

Integration test case I3

Test Case Identifier	I3T1
Test Item(s)	Server Manager → Profile Data Manager
Input Specification	Send Answer
Output Specification	Check if the correct functions are called in Profile Data Manager
Environmental Needs	I2 succeeded

Integration test case I4

Test Case Identifier	I4T1
Test Item(s)	Server Manager → Profile Data Manager
Input Specification	Request to User Manager
Output Specification	Check if the correct functions are called in Profile Data Manager
Environmental Needs	I3 succeeded

Integration test case I5

Test Case Identifier	I5T1
Test Item(s)	User → Call Manager
Input Specification	User Request
Output Specification	Check if the correct functions are called in Call Manager
Environmental Needs	User Driver

Integration test case I6

Test Case Identifier	I6T1
Test Item(s)	Call Manager → Server Manager
Input Specification	Send User Request
Output Specification	Check if the correct functions are called in Server Manager
Environmental Needs	I5 succeeded

Integration test case I7

Test Case Identifier	I7T1
Test Item(s)	Server Manager →Taxi Manager
Input Specification	Request to Taxi Manager
Output Specification	Check if the correct functions are called in Taxi Manager
Environmental Needs	I6 succeeded

Integration test case I8

Test Case Identifier	I8T1
Test Item(s)	Taxi Manager → Server Manager
Input Specification	Taxi Manager Answer
Output Specification	Check if the correct functions are called in Taxi Manager
Environmental Needs	I7 succeeded

Integration test case I9

Test Case Identifier	I9T1
Test Item(s)	User → Cost Estimation Manager
Input Specification	User Request
Output Specification	Check if the correct functions are called in Cost EstimationManager
Environmental Needs	User Driver

Integration test case I10

Test Case Identifier	I10T1
Test Item(s)	Cost Estimation Manager → Server Manager
Input Specification	Send Server Request
Output Specification	Check if the correct functions are called in Server Manager
Environmental Needs	I9 succeeded

Integration test case I11

Test Case Identifier	I11T1
Test Item(s)	Server Manager → Cost Estimation Manager
Input Specification	Send Server Answer
Output Specification	Check if the correct functions are called in Cost EstimationManager
Environmental Needs	I10 succeeded

4.Tools and Test Equipement Required

In this section I will describe two type of testing:

- **Integration Testing** that its aim is to test interfaces and interaction between modules.

- **System Testing** that using a black box strategy its aim is to test functional and non-functional requirements.

For the Integration Testing I use “Arquillian” tool, it is used to execute test cases against the container.

This tool is important to write test for front-end, also using it can write integration test for java EE. To perform this work right is important the interaction with the system.

For System Testing I use “JMeter”, it is used to test logging into application, filling out form, clicking button, link and other components for one or more users.

Also, it can be used to simulate a heavy load on server, network or object to test its strength or to analyze overall performance under different load types.

5.Program Stub and Test Data Required

In this sections I identify all special data that must be tested:

- String composed by different special characters that are used in application forms.
  For example “!£$%&/()=*§°çé_:;”.

- Start and destination that belong or not at the city.

- Negative and positive integers insert in the number of people and number of baggage forms.

  For example “-1,-5”.

- Incorrect format of emails and passwords.

Appendix

The tools we used to create the Code Inspection Document are:

- Microsoft Office Word 2011.

I spent 30 hours to redacting and writing this document.