Test Plan Document

Michele Madaschi Lidia Moioli Luca Martinazzi

January 20, 2016

Contents

Introduction

1.1 Revision History

First version of the ITPD document.

1.2 Purpose and Scope

This document aims to describe, specify and analyze the integration test strategy for *My Taxi Service*, in terms of which components/classes to integrate,the chosen typology of testing and a general schedule to do it, accordingly to what we enstablished in the previous assignments .

1.3 List of Definitions and Abbreviations

1.4 List of Reference Documents

- The project description.
- Our RASD document.
- Ou DD document.

Integration strategy

2.1 Entry criteria

Due to start an integration test two constraints must be satisfied: the major classes must be covered by ,at least , a 60 percent of unit test, while for the others a 30 percent is sufficient.

2.2 Elements to be integrated

In our case element is synonym of class; now we're going to show the classes that need integration test in order to be sure that our application will work correctly.

- Ridesmanager : it needs to be integrated with:
  - Ride, Sharedride : in order to store information about the actived rides
  - Taxiqueue : in order to take information of available taxis in case of taxi request.
  - Controller : in order to exchange information about user's(and also guest's) requests
- Controller : it needs to be integrated with:
  - User : in order to create an ad-hoc Controller and to retrieve information about users
  - Servernetworkinterface : in order to communicate with the corresponding client side
- Servernetworkinterface : it needs to be integrated with:
  - Clientmessage : in order to read client's messages
  - Servermessage : in order to send messages to the client
- Activity : it needs to be integrated with
  - Action : in order to provides the allowed actions
  - Userinterface : in order to provide the set of items this class needs to show

Action	: it needs to be integrated with the Clientnetworkinterface in order to send requests to the server
Userinterface	: it needs to be integrated with the Clientnetworkinterface in order to show the right Activity according to the server message
Clientnetworkinterface	: it needs to be integrated with:

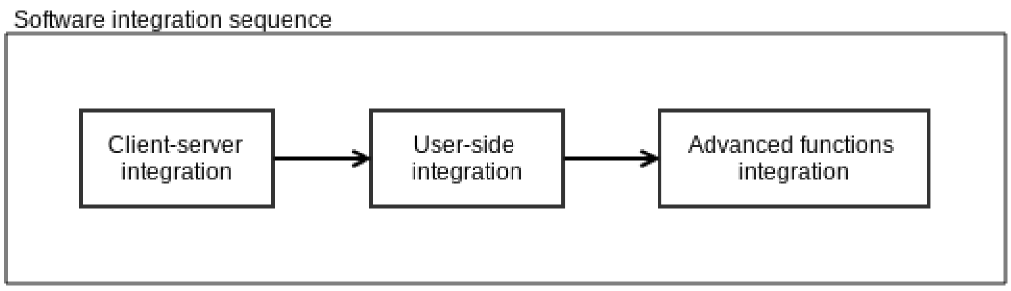
Clientmessage	: in order to send messages to the server
Servermessage	: in order to read server's messages

2.3 Integration testing strategy

In this section we will explain how we plan the integration test in order to build, as soon as possible, a running application with few working features; this will allow us to easly show our progress to the customer, and also , in case of delay, to launch a working application, also with missing requirements. In order to reach our goal we decide to apply a bottom-up method for integration test and top down method for unit test.

2.4 Sequence of component/Function integration

2.4.1 Software integration sequence



Software integration sequence
Client-server integration
User-side integration
Advanced functions integration

Figure 2.1: Software integration

2.4.2 Subsystem integration sequence

The classes are presented here in the ordered sequence in which they will be implemented, which is: 2.2 → 2.3 → 2.4 → 2.5 → 2.6

Note: the arrows here represent the ordering of the implementation, which may happen to partially match the logical structure of the class; however, those arrows do not aim to describe the inter-class relationships

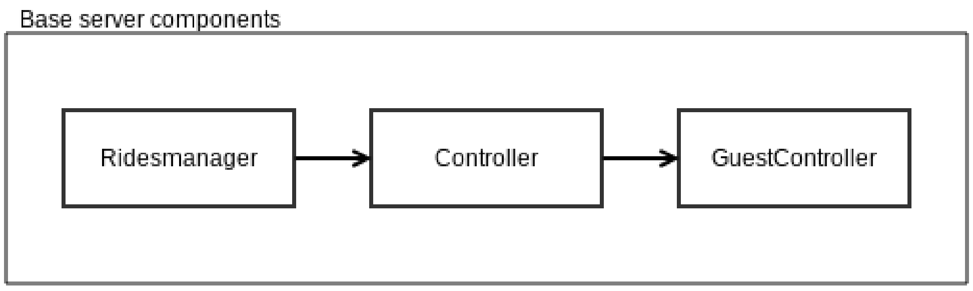


Figure 2.2: Base server components

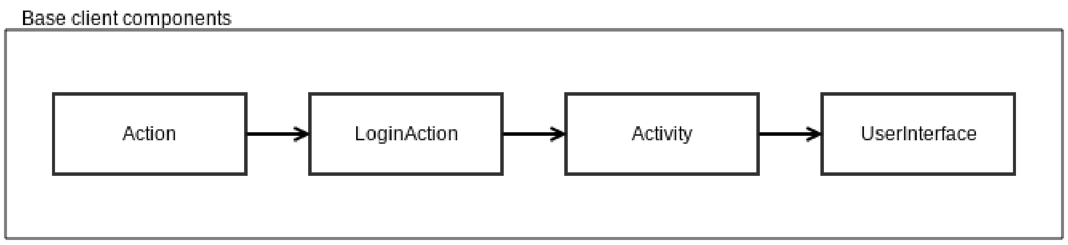


Figure 2.3: Base client components

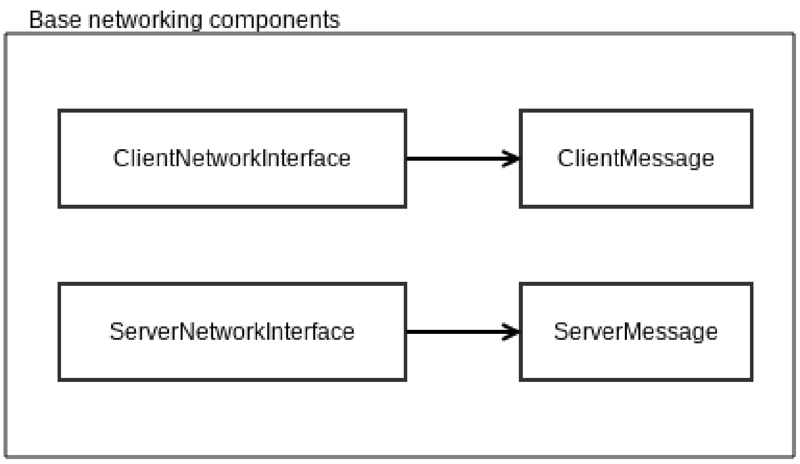


Figure 2.4: Base networking components

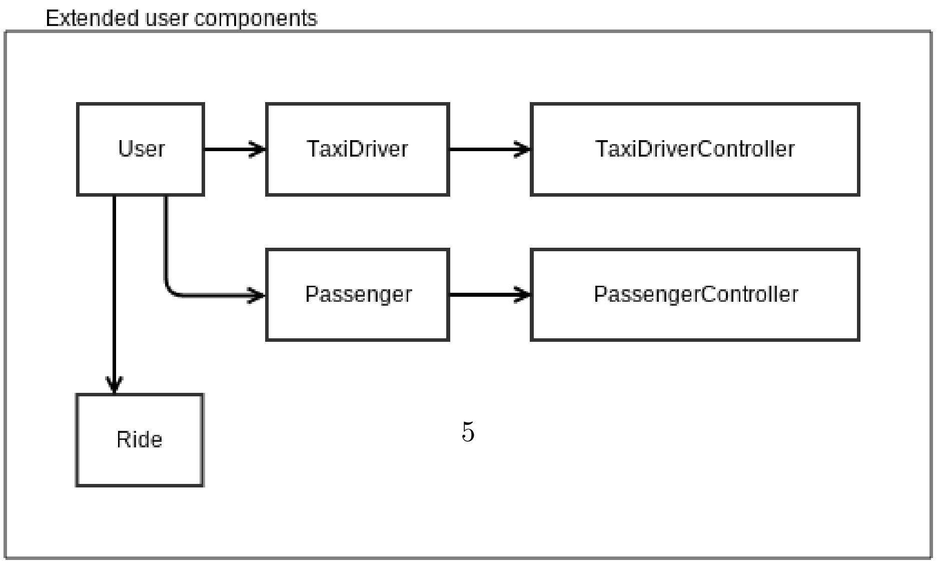


Figure 2.5: Extended client components

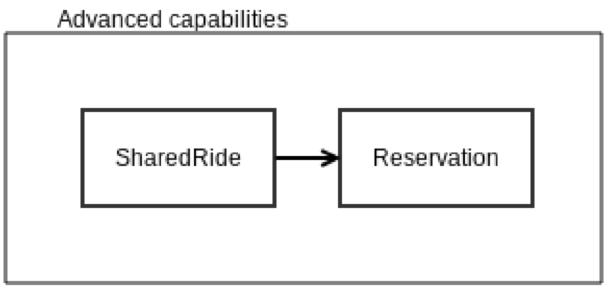


Figure 2.6: Advanced functions

Individual steps and Test description

3.1 Integration test case I-1

Test Case Identifier	I-1-T1
Test Item(s)	Ridesmanager → Controller
Input specification	Create the typical Ridesmanager input
Output specification	Check if the correct methods are called in the Controller
Environmental needs	Ridesmanager driver

3.2 Integration test case I-2

Test Case Identifier	I-2-T1
Test Item(s)	Controller → GuestController
Input specification	Create the typical Controller input
Output specification	Check if the correct methods are called in the GuestController
Environmental needs	I-1 succeeded

3.3 Integration test case I-3

Test Case Identifier	I-3-T1
Test Item(s)	Action → LoginAction
Input specification	??
Output specification	??
Environmental needs	??

3.4 Integration test case I-X4

Test Case Identifier	I-X4-T1
Test Item(s)	ClientNetworkInterface → ClientMessage
Input specification	Invoke various types of network methods
Output specification	Check that the correct ClientMessage(s) are generated
Environmental needs	ClientNetworkInterface driver

3.5 Integration test case I-X5

Test Case Identifier	I-X5-T1
Test Item(s)	ServerNetworkInterface $\rightarrow$ ServerMessage
Input specification	Invoke various types of network methods
Output specification	Check that the correct ServerMessage(s) are generated
Environmental needs	ServerNetworkInterface driver

3.6 Integration test case I-X6

Test Case Identifier	I-X6-T1
Test Item(s)	User $\rightarrow$ Ride
Input specification	Add a new Ride to an User
Output specification	Check that the Ride is correctly added
Environmental needs	User driver

3.7 Integration test case I-X7

Test Case Identifier	I-X7-T1
Test Item(s)	Taxidriver $\rightarrow$ TaxidriversController
Input specification	Add a new Taxidriver to a TaxidriversController
Output specification	Check that the Taxidriver is correctly added
Environmental needs	TaxidriversController driver

3.8 Integration test case I-X8

Test Case Identifier	I-X8-T1
Test Item(s)	Passenger $\rightarrow$ PassengersController
Input specification	Add a new Passenger to a PassengersController
Output specification	Check that the Passenger is correctly added
Environmental needs	PassengersController driver

Tools and testing equipment required

Program stubs and test data required