Politecnico di Milano

Software Engineering II project:
PowerEnjoy

Design Document

Gregori Giacomo and Ruaro Nicola

January 9, 2017
Version 0.1

Contents

Abstract

This document provides a detailed description of the Integration Test's planning for the PowerEnJoy system. It is based on the RASD and DD documents presented in the previous deliveries and must explain to the developement team how to test the system.

Introduction

1.1 Purpose

The purpose of this document is to give a guideline for the developement team in order to effectively test the component's integration. The tests are descibed individually and the required equipment and test-data are listed in the following sections.

1.2 Scope

PowerEnJoy is a car-sharing service based on mobile and web applications which should allow users to reserve vehicles and use them. The application logic must be designed and allocated into components that should improve software maintenability and ease future extensions.

Integration strategy

2.1 Entry criteria

Before proceeding with the integration test in this section we analysed the prerequisites that the software must satisfy.
First of all we must have a code-complete project, all modules must be available and their performances and memory requirements have to fit the specifications.
Secondly all the modules must be unit tested.
Finally the RASD and DD must be completed, they provide all the documentation that we need for proceeding in the succeeding steps.

2.2 Elements to be integrated

In the Design Document, we identified four main Tiers: the EIS Tier, the Business Tier, the Web Tier and the Client Tier. These are the subsystems that we must integrate in this section.
The Enterprise Information System Tier is composed principally by a DBMS that has to be integrated while in the Business Tier all the system components have to be tested individually before to be integrated. The Web Tier relates to the Client Tier and the Business Tier and both the interfaces have to be integrated. Finally, the Client Tier is composed by the On-Board computer, the Mobile application and Web application; they have to be tested individually and then they have to be integrated with their respective Tier.

2.3 Integration testing strategy

We choose for our integration testing strategy to adopt a bottom-up approach. In this way, we test the subsystems from the lower level to the top level, where all the modules are integrated.
There are different advantages following this strategy. The test conditions for each module are easier to create and the test results can be analysed in a simpler way. Then it's easier to localize problems and faults. In the end we can proceed with the test phase of our subsystems alongside their implementation.

On the other side the bottom-up approach brings some disadvantages. The main one is the need of driver programs in order to simulate the missing modules while they aren't already deployed. Another point is the fact that we can't test the whole program until the last module has been developed. Anyway, we think that these disadvantages are bearable comparing the advantages that this approach provides, a last evidence is the fact that probably almost all the faults occurs toward the bottom of the system.
In the testing phase we also selected the order of the subsystems to analyse, not randomly but privileging the critical ones.
We also follow a specific path before performing the integration test. First of all, we design the integration test and the specific drivers if they aren't already done. If it was not made at the unit test we design the input test data, thirdly we set the modules involved, the drivers and the input test data. Finally, we proceed performing the integration test.

2.4 Sequence of Component/Function Integration

NOTE: The structure of this section may vary depending on the integration strategy you select in Section 2.3. Use the structure proposed below as a non mandatory guide.

2.4.1 Software Integration Sequence

The following figures (fig.2.1, fig.2.2) show the components of the PowerEnJoy system integrated into subsystems, the arrows indicate the order of integration.

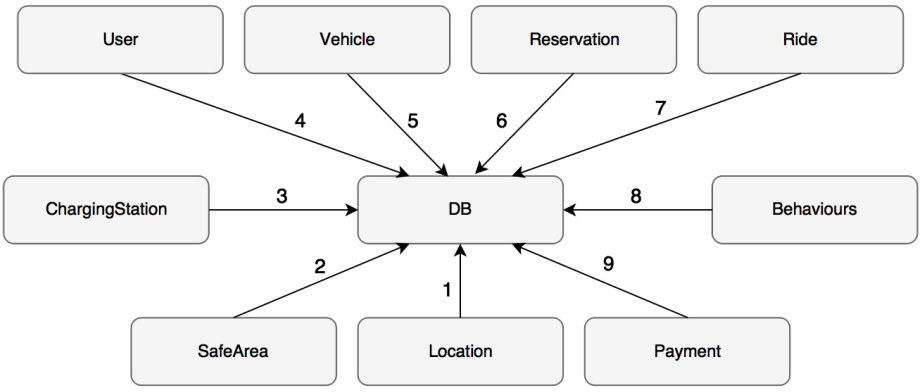

Figure 2.1: Integration sequence for the Enterprise-Information-System subsystem

ID	Integration Test	Paragraphs
I01	Location → DB	3.1 4.1
I02	SafeArea → DB	3.2 4.1
I03	ChargingStation → DB	3.3 4.1
I04	User → DB	3.4 4.1
I05	Vehicle → DB	3.5 4.1
I06	Reservation → DB	3.6 4.1
I07	Ride → DB	3.7 4.1
I08	Behaviours → DB	3.8 4.1
I09	Payment → DB	3.9 4.1

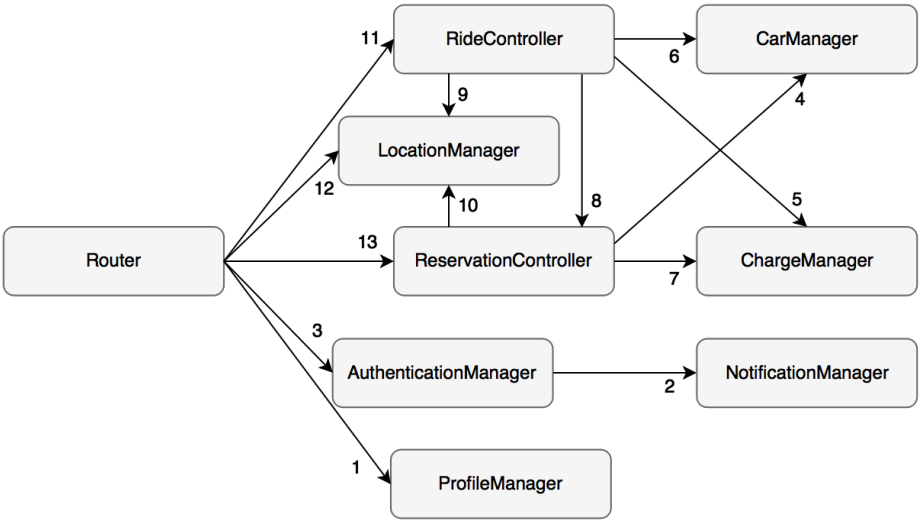

Figure 2.2: Integration sequence for the Business subsystem

ID	Integration Test	Paragraphs
I10	Router → ProfileManager	3.10 4.2
I11	AuthenticationManager → NotificationManager	3.11 4.2
I12	Router → AuthenticationManager	3.12 4.2
I13	ReservationController → CarManager	3.13 4.2
I14	RideController → ChargeManager	3.14 4.2
I15	RideController → CarManager	3.15 4.2
I16	ReservationController → ChargeManager	3.16 4.2
I17	RideController → ReservationController	3.17 4.2
I18	RideController → LocationManager	3.18 4.2
I19	ReservationController → LocationManager	3.19 4.2
I20	Router → RideController	3.20 4.2
I21	Router → LocationManager	3.21 4.2
I22	Router → ReservationController	3.22 4.2

2.4.2 Subsystem Integration Sequence

Figure 2.3 the order in which the subsystems will be integrated.

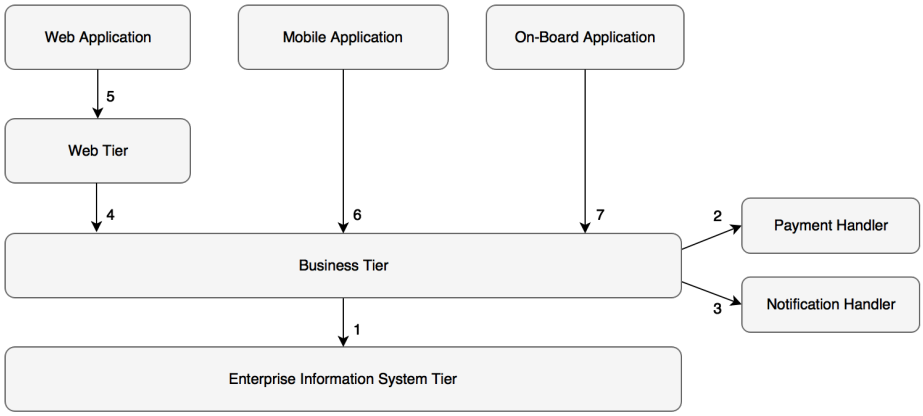

Figure 2.3: Integration sequence for the subsystems

Test case specifications

For each step of the integration process identified above, describe the type of tests that will be used to verify that the elements integrated in this step perform as expected. Describe in general the expected results of the test set. You may refer to Chapter 3 and Chapter 4 of the test plan example [1] as an example of what we expect. (NOTE: This is not a detailed description of test protocols. Think of this as the test design phase. Specific protocols will be written to fulfill the goals of the tests identified in this section.)

3.1 Sample Integration test case I01

Test Case Identifier	I01T1
Test Item(s)	Location → DB
Input Specification	Create typical Location input
Output Specification	Check if the correct functions are called in the DB
Environmental Needs	N/A

3.2 Sample Integration test case I02

Test Case Identifier	I02T1
Test Item(s)	SafeArea → DB
Input Specification	Create typical SafeArea input
Output Specification	Check if the correct functions are called in the DB
Environmental Needs	I1 succeeded

3.3 Sample Integration test case I03

Test Case Identifier	I03T1
Test Item(s)	ChargingStation → DB
Input Specification	Create typical ChargingStation input
Output Specification	Check if the correct functions are called in the DB
Environmental Needs	I1 succeeded

3.4 Sample Integration test case I04

Test Case Identifier	I04T1
Test Item(s)	User → DB
Input Specification	Create typical User input
Output Specification	Check if the correct functions are called in the DB
Environmental Needs	I1 succeeded

3.5 Sample Integration test case I05

Test Case Identifier	I05T1
Test Item(s)	Vehicle → DB
Input Specification	Create typical Vehicle input
Output Specification	Check if the correct functions are called in the DB
Environmental Needs	I1 succeeded

3.6 Sample Integration test case I06

Test Case Identifier	I06T1
Test Item(s)	Reservation → DB
Input Specification	Create typical Reservation input
Output Specification	Check if the correct functions are called in the DB
Environmental Needs	I4 and I5 succeeded

3.7 Sample Integration test case I07

Test Case Identifier	I07T1
Test Item(s)	Ride → DB
Input Specification	Create typical Ride input
Output Specification	Check if the correct functions are called in the DB
Environmental Needs	I1, I4, I5 and I6 succeeded. Payment Driver

3.8 Sample Integration test case I08

Test Case Identifier	I08T1
Test Item(s)	Behaviour → DB
Input Specification	Create typical Behaviour input
Output Specification	Check if the correct functions are called in the DB
Environmental Needs	I7 succeeded

3.9 Sample Integration test case I09

Test Case Identifier	I09T1
Test Item(s)	Payment → DB
Input Specification	Create typical Payment input
Output Specification	Check if the correct functions are called in the DB
Environmental Needs	I4, I6 and I7 succeeded

3.10 Sample Integration test case I10

Test Case Identifier	I10T1
Test Item(s)	Router → ProfileManager
Input Specification	Create typical Router input
Output Specification	Check if the correct functions are called in the ProfileManager
Environmental Needs	N/A

3.11 Sample Integration test case I11

Test Case Identifier	I11T1
Test Item(s)	AuthenticationManager → NotificationManager
Input Specification	Create typical AuthenticationManager input
Output Specification	Check if the correct functions are called in the NotificationManager
Environmental Needs	N/A

3.12 Sample Integration test case I12

Test Case Identifier	I12T1
Test Item(s)	Router → AuthenticationManager
Input Specification	Create typical Router input
Output Specification	Check if the correct functions are called in the AuthenticationManager
Environmental Needs	I11 succeeded

3.13 Sample Integration test case I13

Test Case Identifier	I13T1
Test Item(s)	ReservationController → CarManager
Input Specification	Create typical ReservationController input
Output Specification	Check if the correct functions are called in the CarManager
Environmental Needs	N/A

3.14 Sample Integration test case I14

Test Case Identifier	I14T1
Test Item(s)	RideController → ChargeManager
Input Specification	Create typical RideController input
Output Specification	Check if the correct functions are called in the ChargeManager
Environmental Needs	N/A

3.15 Sample Integration test case I15

Test Case Identifier	I15T1
Test Item(s)	RideController → CarManager
Input Specification	Create typical RideController input
Output Specification	Check if the correct functions are called in the CarManager
Environmental Needs	N/A

3.16 Sample Integration test case I16

Test Case Identifier	I16T1
Test Item(s)	ReservationController → ChargeManager
Input Specification	Create typical ReservationController input
Output Specification	Check if the correct functions are called in the ChargeManager
Environmental Needs	N/A

3.17 Sample Integration test case I17

Test Case Identifier	I17T1
Test Item(s)	RideController → ReservationController
Input Specification	Create typical RideController input
Output Specification	Check if the correct functions are called in the ReservationController
Environmental Needs	I13 and I16 succeeded

3.18 Sample Integration test case I18

Test Case Identifier	I18T1
Test Item(s)	RideController → LocationManager
Input Specification	Create typical RideController input
Output Specification	Check if the correct functions are called in the LocationManager
Environmental Needs	N/A

3.19 Sample Integration test case I19

Test Case Identifier	I19T1
Test Item(s)	ReservationController → LocationManager
Input Specification	Create typical ReservationController input
Output Specification	Check if the correct functions are called in the LocationManager
Environmental Needs	N/A

3.20 Sample Integration test case I20

Test Case Identifier	I20T1
Test Item(s)	Router → RideController
Input Specification	Create typical Router input
Output Specification	Check if the correct functions are called in the RideController
Environmental Needs	I14, I15, I17 and I18 succeeded

3.21 Sample Integration test case I21

Test Case Identifier	I21T1
Test Item(s)	Router → LocationManager
Input Specification	Create typical Router input
Output Specification	Check if the correct functions are called in the LocationManager
Environmental Needs	N/A

3.22 Sample Integration test case I22

Test Case Identifier	I22T1
Test Item(s)	Router → ReservationController
Input Specification	Create typical Router input
Output Specification	Check if the correct functions are called in the ReservationController
Environmental Needs	I13, I16 and I19 succeeded

3.23 Sample Integration test case I23

Test Case Identifier	I22T1
Test Item(s)	Router → ReservationController
Input Specification	Create typical Router input
Output Specification	Check if the correct functions are called in the ReservationController
Environmental Needs	I13, I16 and I19 succeeded

Test procedures

4.1 Sample Integration test procedure TP1

Test Procedure Identifier	TP1
Purpose	This test procedure verifies whether the DB: • can handle entities • can handle client input • can handle agent input • can output requested information to a client • can output requested information to an agent
Procedure Steps	Execute I5-I6 after I1-I4

4.2 Sample Integration test procedure TP2

Test Procedure Identifier	TP1
Purpose	This test procedure verifies whether the DB: • can handle entities • can handle client input • can handle agent input • can output requested information to a client • can output requested information to an agent
Procedure Steps	Execute I5-I6 after I1-I4

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.

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.

Appendix A: Used Tools

A.1 LaTeX

Used to format and redact this document

A.2 *g*it

Used as version control system in order to lead development

A.3 *d*raw.io

Used to draw mockups and diagrams

Appendix B: Hours of work

These are the hours of work spent by each group member in order to redact this document:

- Ruaro Nicola: 2 hours
- Gregori Giacomo: 2 hours
- Total worktime: 4 hours

Appendix C: Revisions

These sections will be eventually redacted during future post-release updates in order to approach the ITPD modifiability providing a comfortable and higly effective way to trace changes:

Bibliography

[1] Luca Mottola and Elisabetta Di Nitto, *Software Engineering 2: Project goal, schedule and rules*, 2016

[2] Nicola Ruaro and Giacomo Gregori, *RASD: Requirements Analysis and Specification Document*, 2016

[3] Nicola Ruaro and Giacomo Gregori, *DD: Design Document*, 2016