Politecnico di Milano

Software Engineering II project:
PowerEnjoy

Integration Test Plan Document

Gregori Giacomo and Ruaro Nicola

January 15, 2017
Version 1.0

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 are going to integrate. Additionally, the Client tier is composed by the Web Application, the Mobile Application and the On-Board Application. Two main external systems, PaymentHandler and NotificationHandler, are to be integrated with the Business Tier.
The subsystems that are to be integrated and their components are exhaustively described in section 2.4 of this document.

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 disadvantage 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

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. Every figure is followed by a table which summarizes the related integration tests.

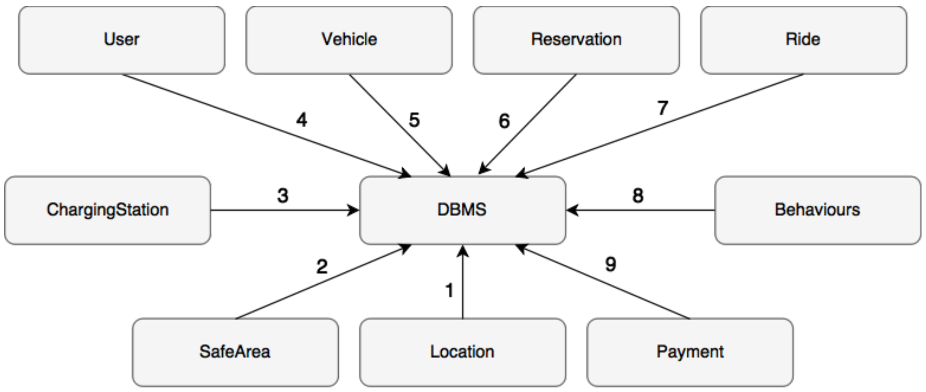

Figure 2.1: Integration sequence for the Enterprise-Information-System subsystem

ID	Integration Test	Paragraphs
I01T1	Location → DBMS	3.1, 4.1
I01T2	SafeArea → DBMS	3.1, 4.1
I01T3	ChargingStation → DBMS	3.1, 4.1
I01T4	User → DBMS	3.1, 4.1
I01T5	Vehicle → DBMS	3.1, 4.1
I01T6	Reservation → DBMS	3.1, 4.1
I01T7	Ride → DBMS	3.1, 4.1
I01T8	Behaviours → DBMS	3.1, 4.1
I01T9	Payment → DBMS	3.1, 4.1

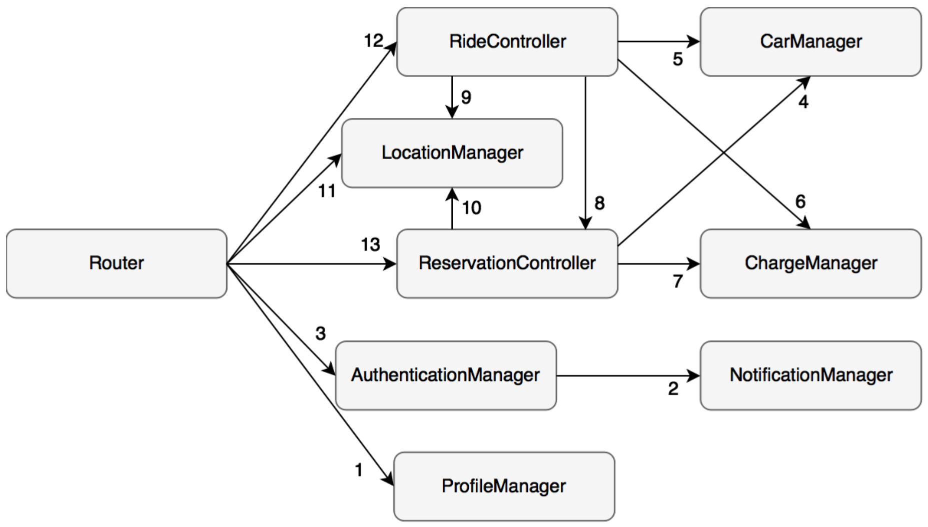

Figure 2.2: Integration sequence for the Business subsystem

ID	**Integration Test**	**Paragraphs**
I02T1	Router → ProfileManager	3.2, 4.2
I03T1	AuthenticationManager → NotificationManager	3.3, 4.2
I04T1	Router → AuthenticationManager	3.4, 4.2
I05T1	ReservationController → CarManager	3.5, 4.2
I05T1	RideController → CarManager	3.5, 4.2
I06T1	RideController → ChargeManager	3.6, 4.2
I06T2	ReservationController → ChargeManager	3.6, 4.2
I07T1	RideController → ReservationController	3.7, 4.2
I08T1	RideController → LocationManager	3.8, 4.2
I08T2	ReservationController → LocationManager	3.8, 4.2
I08T3	Router → LocationManager	3.8, 4.2
I09T1	Router → RideController	3.9, 4.2
I10T1	Router → ReservationController	3.10, 4.2

2.4.2 Subsystem Integration Sequence

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

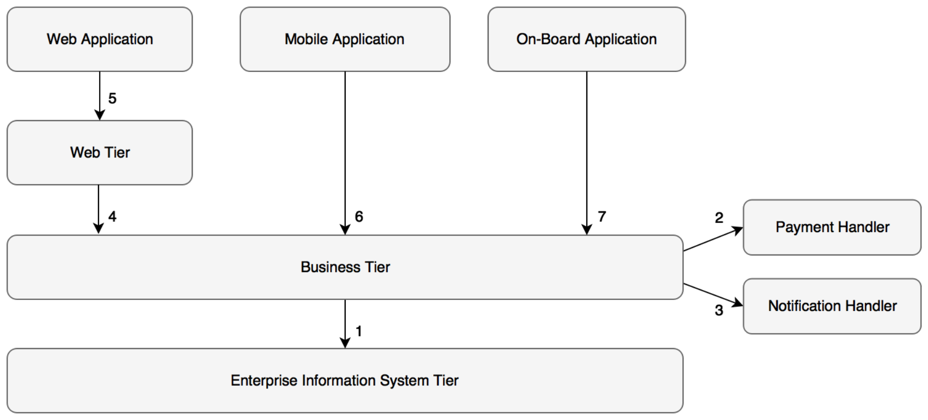

Figure 2.3: Integration sequence for the subsystems

ID	**Integration Test**	**Paragraphs**
I11T1	Business Tier → Enterprise Information System Tier	3.11, 4.3
I12T1	Business Tier → Payment Handler	3.12, 4.3
I13T1	Business Tier → Notification Handler	3.13, 4.3
I14T1	Web Tier → Business Tier	3.14, 4.3
I15T1	Web Application → Web Tier	3.15, 4.3
I16T1	Mobile Application → Business Tier	3.16, 4.3
I17T1	On-Board Application → Business Tier	3.17, 4.3

Test case specifications

In this chapter are described the tests used in order to achieve our integration strategy and to verify that the integrated elements perform as expected.

3.1 Integration test case I01

In the following tables of section 3.1 the integrations among the Entity Beans and the DBMS are described.

Test Case Identifier	I01T1
Test Item(s)	Location → DBMS
Input Specification	Generate typical queries for the Location table
Output Specification	Check the correctness of the DBMS answer
Environmental Needs	N/A

Test Case Identifier	I01T2
Test Item(s)	SafeArea → DBMS
Input Specification	Generate typical queries for the SafeArea table
Output Specification	Check the correctness of the DBMS answer
Environmental Needs	I01T1 succeeded

Test Case Identifier	I01T3
Test Item(s)	ChargingStation → DBMS
Input Specification	Generate typical queries for the ChargingStation table
Output Specification	Check the correctness of the DBMS answer
Environmental Needs	I01T1 succeeded

Test Case Identifier	I01T4
Test Item(s)	User $\rightarrow$ DBMS
Input Specification	Generate typical queries for the User table
Output Specification	Check the correctness of the DBMS answer
Environmental Needs	I01T1 succeeded

Test Case Identifier	I01T5
Test Item(s)	Vehicle $\rightarrow$ DBMS
Input Specification	Generate typical queries for the Vehicle table
Output Specification	Check the correctness of the DBMS answer
Environmental Needs	I01T1 succeeded

Test Case Identifier	I01T6
Test Item(s)	Reservation $\rightarrow$ DBMS
Input Specification	Generate typical queries for the Reservation table
Output Specification	Check the correctness of the DBMS answer
Environmental Needs	I01T4 and I01T5 succeeded

Test Case Identifier	I01T7
Test Item(s)	Ride $\rightarrow$ DBMS
Input Specification	Generate typical queries for the Ride table
Output Specification	Check the correctness of the DBMS answer
Environmental Needs	I01T1, I01T4, I01T5 and I01T6 succeeded. Payment Driver

Test Case Identifier	I01T8
Test Item(s)	Behaviour $\rightarrow$ DBMS
Input Specification	Generate typical queries for the Behaviour table
Output Specification	Check the correctness of the DBMS answer
Environmental Needs	I01T7 succeeded

Test Case Identifier	I01T9
Test Item(s)	Payment → DBMS
Input Specification	Generate typical queries for the Payment table
Output Specification	Check the correctness of the DBMS answer
Environmental Needs	I01T4, I01T6 and I01T7 succeeded

3.2 Integration test case I02

Test Case Identifier	I02T1
Test Item(s)	Router → ProfileManager
Input Specification	Generate typical Router input for the ProfileManager component
Output Specification	Check if the correct methods are called and the computations are performed correctly in the ProfileManager
Environmental Needs	N/A

3.3 Integration test case I03

Test Case Identifier	I03T1
Test Item(s)	AuthenticationManager → NotificationManager
Input Specification	Generate typical AuthenticationManager input
Output Specification	Check if the correct methods are called and the computations are performed correctly in the NotificationManager
Environmental Needs	N/A

3.4 Integration test case I04

Test Case Identifier	I04T1
Test Item(s)	Router → AuthenticationManager
Input Specification	Generate typical Router input for the AuthenticationManager component
Output Specification	Check if the correct methods are called and the computations are performed correctly in the AuthenticationManager
Environmental Needs	I03T1 succeeded

3.5 Integration test case I05

Test Case Identifier	I05T1
Test Item(s)	ReservationController → CarManager
Input Specification	Generate typical ReservationController input
Output Specification	Check if the correct methods are called and the computations are performed correctly in the CarManager
Environmental Needs	N/A

Test Case Identifier	I05T2
Test Item(s)	RideController → CarManager
Input Specification	Generate typical RideController input
Output Specification	Check if the correct methods are called and the computations are performed correctly in the CarManager
Environmental Needs	N/A

3.6 Integration test case I06

Test Case Identifier	I06T1
Test Item(s)	RideController → ChargeManager
Input Specification	Generate typical RideController input
Output Specification	Check if the correct methods are called and the computations are performed correctly in the ChargeManager
Environmental Needs	N/A

Test Case Identifier	I06T2
Test Item(s)	ReservationController $\rightarrow$ ChargeManager
Input Specification	Generate typical ReservationController input
Output Specification	Check if the correct methods are called and the computations are performed correctly in the ChargeManager
Environmental Needs	N/A

3.7 Integration test case I07

Test Case Identifier	I07T1
Test Item(s)	RideController $\rightarrow$ ReservationController
Input Specification	Generate typical RideController input
Output Specification	Check if the correct methods are called and the computations are performed correctly in the ReservationController
Environmental Needs	I05T1 and I06T2 succeeded

3.8 Integration test case I08

Test Case Identifier	I08T1
Test Item(s)	RideController $\rightarrow$ LocationManager
Input Specification	Generate typical RideController input
Output Specification	Check if the correct methods are called and the computations are performed correctly in the LocationManager
Environmental Needs	N/A

Test Case Identifier	I08T2
Test Item(s)	ReservationController $\rightarrow$ LocationManager
Input Specification	Generate typical ReservationController input
Output Specification	Check if the correct methods are called and the computations are performed correctly in the LocationManager
Environmental Needs	N/A

Test Case Identifier	I08T3
Test Item(s)	Router $\rightarrow$ LocationManager
Input Specification	Generate typical Router input for the LocationManager component
Output Specification	Check if the correct methods are called and the computations are performed correctly in the LocationManager
Environmental Needs	N/A

3.9 Integration test case I09

Test Case Identifier	I09T1
Test Item(s)	Router $\rightarrow$ RideController
Input Specification	Generate typical Router input for the RideController component
Output Specification	Check if the correct methods are called and the computations are performed correctly in the RideController
Environmental Needs	I05T2, I06T1, I07T1 and I08T1 succeeded

3.10 Integration test case I10

Test Case Identifier	I10T1
Test Item(s)	Router $\rightarrow$ ReservationController
Input Specification	Generate typical Router input for the ReservationController component
Output Specification	Check if the correct methods are called and the computations are performed correctly in the ReservationController
Environmental Needs	I05T1, I06T2 and I08T2 succeeded

3.11 Integration test case I11

Test Case Identifier	I11T1
Test Item(s)	Business Tier → Enterprise Information System Tier
Input Specification	Generate typical Business Tier input
Output Specification	Check if the correct methods are called and the computations are performed correctly in the Enterprise Information System Tier
Environmental Needs	N/A

3.12 Integration test case I12

Test Case Identifier	I12T1
Test Item(s)	Business Tier → Payment Handler
Input Specification	Generate typical Business Tier input
Output Specification	Check if the correct methods are called and the computations are performed correctly in the Payment Handler
Test description	In particular it's the ChargeManager, located in the Business Tier subsystem, that interacts with the Payment Handler
Environmental Needs	Payment Handler Stub

3.13 Integration test case I13

Test Case Identifier	I13T1
Test Item(s)	Business Tier → Notification Handler
Input Specification	Generate typical Business Tier input
Output Specification	Check if the correct methods are called and the computations are performed correctly in the Notification Handler
Test description	In particular it's the NotificationManager, located in the Business Tier subsystem, that interacts with the Notification Handler
Environmental Needs	Notification Handler Stub

3.14 Integration test case I14

Test Case Identifier	I14T1
Test Item(s)	Web Tier → Business Tier
Input Specification	Generate typical Web Tier input
Output Specification	Check if the correct methods are called and the computations are performed correctly in the Business Tier
Environmental Needs	I11T1, I12T1 and I13T1 succeeded

3.15 Integration test case I15

Test Case Identifier	I15T1
Test Item(s)	Web Application → Web Tier
Input Specification	Generate typical Web Application input
Output Specification	Check if the correct methods are called and the computations are performed correctly in the Web Tier
Environmental Needs	I14T1 succeeded

3.16 Integration test case I16

Test Case Identifier	I16T1
Test Item(s)	Mobile Application → Business Tier
Input Specification	Generate typical Mobile Application input
Output Specification	Check if the correct methods are called and the computations are performed correctly in the Business Tier
Environmental Needs	I11T1, I12T1 and I13T1 succeeded

3.17 Integration test case I17

Test Case Identifier	I17T1
Test Item(s)	On-Board Application $\rightarrow$ Business Tier
Input Specification	Generate typical On-Board Application input
Output Specification	Check if the correct methods are called and the computations are performed correctly in the Business Tier
Environmental Needs	I11T1, I12T1 and I13T1 succeeded

Test procedures

4.1 Sample Integration test procedure TP1

Test Procedure Identifier	TP1
Purpose	This test procedure verifies whether the DBMS: • can handle entities input • can execute correctly all the queries and output coherent response messages
Procedure Steps	Execute the I01 from T1 to T9 following the order

4.2 Sample Integration test procedure TP2

Test Procedure Identifier	TP2
Purpose	This test procedure verifies whether the Business Tier: • can handle client tier input • can handle interactions among components • can output requested information to the Notification Handler • can output requested information to the Payment Handler • can output requested information to the Enterprise Information System Tier
Procedure Steps	Execute from I02T1 to I10T1 following the order

4.3 Sample Integration test procedure TP3

Test Procedure Identifier	TP3
Purpose	This test procedure verifies whether PowerEnJoy: • can handle Web Application input • can handle Mobile Application input • can handle On-Board Application input • can handle interactions among subsystems • can output requested information to the Notification Handler • can output requested information to the Payment Handler • can output requested information to the Enterprise Information System Tier
Procedure Steps	Execute from I11T1 to I17T1 following the order

Tools and test equipment required

5.1 Arquillian

Arquillian (*arquillian.org*) is a flexible and portable integration testing framework designed specifically for JEE. It will be used to test the correct behaviour of the containers and their interaction with the system.

5.2 JMeter

JMeter (*jmeter.apache.org*) is an open-source application developed and maintained by the Apache foundation. It is designed to load test functional behavior and measure performance: in this project it will be used to achieve system and non-functional requirements testing.

5.3 JUnit

JUnit (*junit.org*) is probably the most common framework for Java unit testing. In this project, though, it will be used in combination with Arquillian and Mockito to achieve High-level and integration testing.

5.4 Mockito

Mockito (*site.mockito.org*) is a clean, simple and well supported Java mocking framework. It will be used to verify the interactions between objects and to create stubs when a componet cannot be tested in isolation.

5.5 Test equipment

The PowerEnJoy application needs to be deployed on different machines. Precisely, two main machines are needed: one machine capable of running an instance of the GlassFish Server and one machine capable of running the DBMS.

Program stubs and test data required

6.1 Drivers

For the PowerEnJoy system integration tests we decided to use a bottom-up strategy, this approach requires drivers in order to simulate components and invoke methods on the under-integration component.

The required drivers are:

Entity Bean Drivers: these drivers are used to invoke the methods exposed by the Entity Java Beans components in order to test their integration with the connected components

RouterDriver: this driver is used to invoke the methods exposed by the Router component in order to test its integration with the connected components

RideControllerDriver: this driver is used to invoke the methods exposed by the RideController component in order to test its integration with the connected components

ReservationControllerDriver: this driver is used to invoke the methods exposed by the ReservationController component in order to test its integration with the connected components

AuthenticationManagerDriver: this driver is used to invoke the methods exposed by the AuthenticationManager component in order to test its integration with the connected components

6.2 Stub

Payment Handler Stub: the interactions with the Payment Handler must be tested using a stub in order to verify that all the calls performed over the provided API are correctly formed.

Notification Handler Stub: the interactions with the Notification Handler must be tested using a stub in order to verify that all the calls performed over the provided API are correctly formed.

6.3 Data

DataBase: the Entity Java Beans components must be integrated with the DBMS, therefore a "Testing" DB with the same structure of the production's DB is needed. Such DataBase must have enough entries to allow extensive testing of the Enterprise Information System.

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: 17 hours
- Gregori Giacomo: 17 hours
- Total worktime: 34 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:

Glossary

Enterprise Java Bean Enterprise Java Beans (EJB) is a development architecture for building highly scalable and robust enterprise level applications to be deployed on J2EE compliant Application Server.

Acronyms

DB	Database.
DBMS	Database management system.
DD	Design Document.
EIS	Enterprise Information System.
EJB	Enterprise Java Beans.
ER	Entity-Relationship Diagram: diagram that shows the relationships of entity sets stored in a database..
ITPD	Integration Test Plan Document.
JEB	Java Entity Bean.
RASD	Requirements Analysis and Specification Document.
RDBMS	Relational database management system.
SQL	Structured Query Language.

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