POLITECNICO
MILANO 1863

Software Engineering 2 project: *PowerEnJoy*

A.A. 2016/2017 - Professor E. di Nitto

Integration Test Plan Document

Version **1.0** –15/01/2018

Pietro Avolio Mat 878640
Guido Borrelli Mat 874451

Contents

1 Introduction **3**
- 1.1 Purpose and scope . . . 3
- 1.2 Definitions, acronyms and abbreviations . . . 3
- 1.3 Reference Documents: . . . 3
- 1.4 Document Structure . . . 3

2 Integration Strategy **5**
- 2.1 Entry criteria . . . 5
- 2.2 Elements to be integrated . . . 5
- 2.3 Integration testing strategy . . . 6
- 2.4 Integration sequence . . . 6
  - 2.4.1 Sub-components of the core component integration sequence 6
  - 2.4.2 Components integration sequence . . . 6

3 Test Description **7**
- 3.1 Test case specification . . . 7
- 3.2 Test procedures . . . 7

4 Tools and Test Equipment Required **7**

5 Program Stubs and Test Data Required **7**

6 Appendix **8**
- 6.1 Tools used . . . 8
- 6.2 Effort spent . . . 8

1 Introduction

1.1 Purpose and scope

The purpose of this integration test plan document is to provide information about the integration test activities to be performed for the PowerEnJoy project. This includes how to accomplish the integration tests, which approach to follow and which tools to use.

The aim of the integration test activities is to verify that the different components of the system interoperate coherently in order to fulfill the functional and non-functional requirements of the system.
Since some components are made up of several other sub-components, the integration tests must be performed inside those components too.

1.2 Definitions, acronyms and abbreviations

In order to avoid ambiguity and possible misunderstandings, this table is an integration to the definitions, the acronyms and the abbreviations defined in the RASD and DD documents.

RASD	Requirements Analysis and Specication Document
DD	Design document
ITPD	Integration Test Plan Document
Component	A part of the system implementing high-level functionalities
Sub-Component	A part of a component
LOF	Lines Of Code

1.3 Reference Documents:

- Integration Test Plan Documents of the previous years published on BeeP.
- spinGRID integration test plan on BeeP.
- Testing slides on BeeP .
- Tips and Trick for Supporting Architecture Description with UML on BeeP.
- Integration Testing page on Wikipedia (visited 08/01/2017).

1.4 Document Structure

The document is structured in the following chapters:

Introduction This chapter provides preliminary informations about the document.

Integratio Strategy This chapter provides detailed information about the entry criteria to be met before starting the integration test, the elements to be integrated and an outline of the main strategies to follow.

Individual Steps and Test Description This chapter provides .

Tools and Test Equipment Required This chapter provides.

Program Stubs and Test Data Required This chapter shows how.

2 Integration Strategy

2.1 Entry criteria

In this section we describe the prerequisites needed before the integration tests can be started.

Methods and classes should be locally unit tested and achieve at least an 80% LOF coverage in order to discover major issues in the effective implementation of the algorithms and in the usage of the data structures. Unit tests should be performed during the development.

The project should also have passed through a phase of code inspection in order to ensure maintenability, the respect of the convenentions, usage of the known best practices and the avoidance of the bad ones. The code inspection should be automated via automated tools as much as possibile in order to be able to spent manual inspection on the most complex parts.

The following documents describing functional and non functional requirements, behaviour and purpose of the system must be delivered:

- Requirements Analysis and Specication Document (RASD).
- Design Document (DD).
- Integration Test Plan Document (this document).

Finally, an up-to-date and complete documentation of the code, using a common paradigm such as JavaDoc, could be a solid base and a good reference for the integration tests development.

2.2 Elements to be integrated

As stated in the DD document, the system is splitted on four logical layers, each of those containing one or more components:

Client Layer This logical layer contains the PowerEnJoy Mobile Application, the PowerEnJoy Control Room and the vehicles.

Web Server This logical layer contains the web server required to host the PowerEnJoy Control Room.

Business Tier This logical layer contains the PowerEnJoy Core component.

Data Tier This logical layer contains the RDBS and the NoSqlDBMS.

These logical layers and the components they contain communicate using the interfaces described in the DD, which are object of the integration tests.

The PowerEnJoy Core component, which contains all the application logic, is made up of several sub-components grouped in various containers as described in the DD document; it also communicates with some external gateways. The way these sub-components integrate one with the other and the way they communicate with the external gateways is the main part of the integration test activities to be performed.

2.3 Integration testing strategy

In order to prevent all the possibile issues coming from a typical big bang approach, in which all the integration tests are performed at the end of the development process, a slowly incremental approach is the one chosen for the integration tests of the system. The incremental approach facilitates the localization and the recognition of the flaws in the code.

To limitate the number of stubs and mockups to be realized, the integration tests should follow a bottom-up flow, starting from the smallest - and oftem more critical - sub-components of the PowerEnJoy Core to the biggest ones.
The last tests to be perfomed are the ones concerning the intercommunication betweent components, which is mainly based on RESTful APIs and HTTPs protocol.

2.4 Integration sequence

In this section are descripbed the sequences in which both the high-level components and the sub-components should be integrated. The integration tests must proced from the sub-components to the components ones.

2.4.1 Sub-components of the core component integration sequence

The sub-components integration tests sequence is described in table 1 and figure 1. As mentioned in section 2.3, the sequence moves from the most independent to the less independent component.

Table 1: Integration of the sub-components of the core component

Figure 1: Sequence diagram of the integration of the sub-components of the core component

2.4.2 Components integration sequence

The integration testes sequence of the main components of the system is described in table 2and figure 2.

Table 2: Integration of the main components of the system

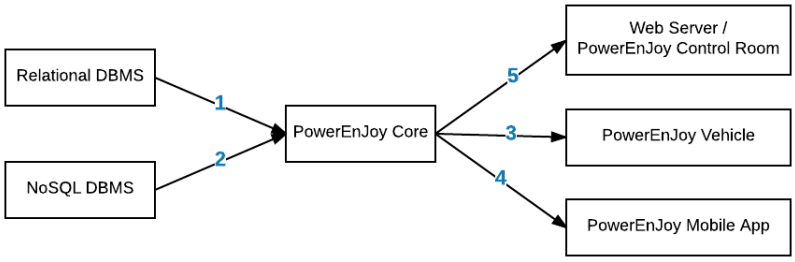

Figure 2: Sequence diagram of the integration of the main components

3 Test Description

3.1 Test case specification

Here we will schematizes and detail every integration steps of the integration sequence identified in section 2.4.

Una volta decisa la sequenza di integrazione, x ogni step dobbiamo fare una test case specification

3.2 Test procedures

Dobbiamo verificare con particolare attenzione alcune procedure (tipo: prenotazione, noleggio, termine noleggio, apertura auto) utilizzando come steps i test case precedenti.

4 Tools and Test Equipment Required

The following software here mentioned are identified to assist the integration phase of all various components of PowerEnJoy core software.

- Mockito: Let us mocking a persistence manager, an external service or an unavailable class. So, it permit to check the validity of interaction and data coherency in a fast way, in fact we don't need to setup our DB and no environments.
- Arquillian: Let us make integration tests against containers, in such a way we can check interaction with the system and isolate during the tests a small group of classes and resources needed.
- JUnit: non è menzionato nei testing tools delle slide, però tutti ne parlano e lo mettono. Però non bisogna confondere con il unit test perché è un'altra frase del testing, non questa. No?

5 Program Stubs and Test Data Required

Program stubs non ho capito bene cosa siano.

Test data required direi che è fondamentale testare la correttezza della 'zona verde' di noleggio sicuramente.

Poi la correttezza dei dati ricevuti e inviati alle auto, perché è uno scambio "critico".

Altro non mi viene in mente ora, però non ho ancora avuto l'ispirazione sul come metterlo giù anche perché devo capire ancora benino la parte degli stubs.

6 Appendix

6.1 Tools used

- LyX as LaTeX editor.
- Lucidchart for diagrams.
- Github as version controller.

6.2 Effort spent

Pietro Avolio	
Guido Borrelli	