Integration Test Plan Document - Software Engineering 2

POLITECNICO
MILANO 1863

PowerEnjoy

Marini Alberto
862838
alberto2.marini@mail.polimi.it

Marrone Matteo
810840
matteo.marrone@mail.polimi.it

Sabatelli Antonella
875666
antonella.sabatelli@mail.polimi.it

January 15th, 2017

Politecnico di Milano

Contents

Chapter 1

1 Introduction

1.1 Purpose and Scope

This document is the Integration **Test Plan Document (ITPD)** for our *PowerEnjoy* app. Its purpose is to determine how to accomplish the integration test of the software, which tools are to be used and which approach will be followed.

PowerEnjoy is an app for electric car sharing in Milan area. Its goal is to simplify the ability of reserving and using a car. *PowerEnjoy* guarantees a smart fare management, by respecting the service rules.

1.2 List of Definitions and Abbreviations

- **RASD**: Requirements Analysis and Specification Document.
- **DD**: Design Document.
- **ITPD**: Integration Test Plan Document.
- **DB**: DataBase.
- **DBMS**: DataBase Manager System.

1.3 List of Reference Documents

- Requirements Analysis and Specification Document (RASD) PowerEnjoy - *Marini, Marrone, Sabatelli*
- Design Document (DD) PowerEnjoy - *Marini, Marrone, Sabatelli*

Chapter 2

2 Integration Strategy

2.1 Entry Criteria

Before integration testing can start, the project should be code-complete, that is, devoid of any missing parts, and JUnit tests should have been developed for every module of the software, possibly covering 85% or more of the lines of code and making use of standard white-box testing approaches such as path testing; latest releases of documentation to be used as reference for the integration testing phase (see References section), including this document, should be made available to the developers, as well.

It is assumed that the integration testing will be carried out only after that all testing tools referenced in *Chapter 4* will have been properly installed and configured, making use of hardware components possessing the characteristics described in the Design Document (see References section).

2.2 Elements to be Integrated

The integration process will take place on two levels: the components one and the subsystems one. The main components whose functionality we are interested in testing are the Enterprise Java Beans defining the business logic within the second tier of the system, the one containing the application server that well mark as a subsystem. Well focus in particular on the User Manager, the Search Manager, the Reservation Manager, the Ride Manager and the Communication Manager. As mentioned in the Design Document, the data tier and the car software will be acquired from external agents, with the exclusion of some customisations of the car software added ad hoc, yet they are to be integrated and will be marked as two different subsystems as well. The client tier, where the GUI is located, will be the final subsystem to integrate.

2.3 Integration Testing Strategy

Well make us of a bottom-up approach, beginning from the integration of the smallest core components of the business logic to then proceed with the integration of the subsystem containing them with the other subsystems previously mentioned.

This approach will prove more advantageous than a top-down one since we believe the development of stubs mimicking the modules not yet integrated would be more complex than the one of the driver mocking the skeletal structure of the system, and that the observation of the test results would be easier as well.

Finally, in our experience, bugs are more easily located as well as fixed when the modules containing them are considered *per se*, rather than when they are already attached to a system in the course of development.

2.4 Sequence of Component/Function Integration

2.4.1 Software Integration Sequence

2.4.2 Subsystem Integration Sequence

Chapter 3

3 Individual Steps and Test Description

Chapter 4

4 Tools and Test Equipment Required

Chapter 5

5 Program Stubs and Test Data Required

Chapter 6

6 Effort Spent