PowerEnJoy
Software Engineering II

Integration Test Plan Document

Giovanni Scotti, Marco Trabucchi

Document version: 1
December 28, 2016

Contents

Section 1

Introduction

1.1 Purpose and Scope

1.1.1 Purpose

The Integration Test Plan Document (ITPD) is intended to provide the guidelines to accomplish the integration test phase planning in sufficient detail. This also includes determining which tools are needed and will be used during the testing process itself, as well as the required stubs, drivers and data structures that will be useful during said process.

1.1.2 Scope

PowerEnJoy is a car sharing service that only employs electric vehicles; it is provided for a large city, and aims to support the sharing process and car management of the electric cars, as well as the booking and payments for the service itself.

1.2 Definitions, Acronyms, Abbreviations

RASD: Requirements and Specification Document.

DD: Design Document.

ITPD: Integration Test Plan Document.

1.3 Reference Documents

The indications provided in this document are based on the ones stated in the previous deliverables for the project, the RASD document [1] and the DD document [2].

Moreover it is strictly based on the test plan example [4] presented during lectures and on the specifications concerning the RASD assignment [3] for the Software Engineering II project, part of the course held by professors Luca Mottola and Elisabetta Di Nitto at the Politecnico di Milano, A.Y. 2016/17.

Section 2

Integration Strategy

2.1 Entry Criteria

This section expresses the prerequisites needed to be met before the integration phase takes place.

Documentation: The documentation for every method and class must be provided for each individual component, in order to make it easier to reuse classes and understand their functioning; this is in fact also a prerequisite for the unit tests to be performed before the integration test phase. When necessary, a formal language specification of the classes' behaviours can be used (such as JML - Java Modelling Language).

Unit tests: All the classes and methods must be tested thoroughly using JUnit, in order to assure a properly correct behaviour of the internal mechanics of the individual components. It is required that the test coverage of each class and package reaches 90% of the code lines; moreover, test cases must be written with continuity and executed at every consecutive build of the project: this is needed in order to ensure that newly added lines do not interfere with the stability of the rest of the code.

Code Inspection and Analysis: Both automated data-flow analysis and code inspection must be performed on the whole project classes. This will reduce the risk that, during the integration test phase, any code-related issues or bugs rise, leading to more complex problematic situations to be solved in latter phases of the project development, with much greater effort for the development team.

RASD and DD: Along with the indications provided in this very document (ITPD), the two previous documents for this project, RASD and DD, must be delivered before the integration test phase can begin.

2.2 Elements to be Integrated

The integration test phase for the *PowerEnJoy* system will be structured on the architectural division in tiers that is described in the Design Document [2], as well as the indication of the elements of which said subsystems are composed of.

With respect to this, the subsystems to be integrated in this phase are the following four:

Database Tier This includes all the commercial database structures that will be used for the data storage and management of the system, namely the DBMS and the Database Engine; the two data layer components are already developed to work properly when coupled together, so the only component to be integrated is the DBMS.

Application Logic Tier This includes all the business logic for the application, the data access components and the interface components towards external systems and clients. All the interactions among internal logic components must be tested and all the subsystems that interact with this tier must be individually integrated.

Web Tier This includes all the components in charge of the web interface and the communication with the application logic tier and the browser client. The integration tests must be performed both ways for this tier, and the Web Controller must be thoroughly tested also for the interaction with the Java Server Pages component.

Client Tier This includes the various types of clients, which is to say the Mobile Application Client, the Web Browser Client and the On-Board Application Client, and their internal components. Single clients must behave properly with respect to their internal structure, and must be individually integrated with the tier they interface with.

The integration process will be performed in two steps:

- A *first phase* in which the individual components of the subsystems (i.e. Java classes, Java Beans and Containers), are integrated one by one.

- A *second phase* in which, after having ensured a proper internal behaviour, the above specified subsystems are integrated as well.

2.3 Integration Testing Strategy

As far as the integration testing process is concerned, a **bottom-up approach** will be followed.

The choice of the bottom-up testing strategy is natural since the integration testing can start from the smallest and lowest-level components, that already have unit tests and do not depend on other components or not-already-developed components. In this way the total amount of needed stubs to accomplish the integration will be deeply reduced, but temporary programs for higher-level modules (drivers) are used to simulate them and invoke the unit under test.

The bottom-up strategy will be mixed with a **critical-module-first approach**, in order to avoid issues related to the failures of core components and to pose a threat to the correct implementation of the entire *PowerEnJoy* system.

Moreover the higher-level subsystems described in section 2.2 are loosely coupled and fairly independent from one another because they correspond to different tiers. In this case, the critical-module-first approach is used to establish the integration order to obtain the full system.

Notice that the DBMS is a commercial component already developed that can be used directly in the bottom-up approach and does not have any dependency.

2.4 Sequence of Components/Function Integration

The following sections aims to describe the integration and integration testing sequence of the different components and subsystems of *PowerEnJoy*. From now on the following notation will be used: an arrow going from component C1 to component C2 points out that C2 is necessary to C1 to work properly.

2.4.1 Software Integration Sequence

The components of each subsystem are tested starting from the most to the least independent one.

2.4.2 Subsystem Integration Sequence

The integration sequence of the high-level subsystems is described in Figure 2.1 and Table 2.1.

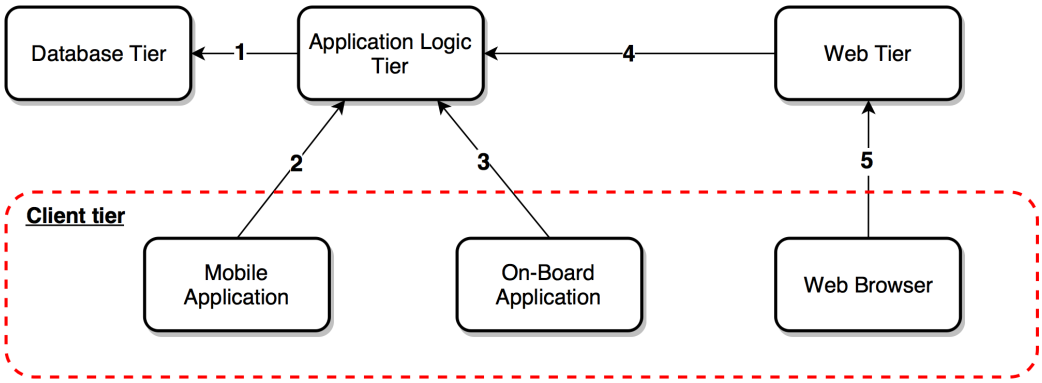

Figure 2.1: Diagram representing the order of the subsystems integration.

N.	Subsystem	Integrates with
SI1	Application Logic Tier	Database Tier
SI2	Mobile Application	Application Logic Tier
SI3	On-Board Application	Application Logic Tier
SI4	Web Tier	Application Logic Tier
SI5	Web Browser	Web Tier

Table 2.1: Integration order of the subsystems described in Section 2.2.

Section 3

Individual Steps and Test Description

Section 4

Tools and Test Equipment Required

Section 5

Program Stubs and Test Data Required

Appendix A

Appendix

A.1 Software and tools used

- LaTeX, used as typesetting system to build this document.
- draw.io - `https://www.draw.io` - used to draw diagrams and mock-ups.
- GitHub - `https://github.com` - used to manage the different versions of the document and to make the distributed work much easier.
- GitHub Desktop, the GitHub official application that offers a seamless way to contribute to projects.

A.2 Hours of work

The absolute major part of the document was produced in group work. The approximate number of hours of work for each member of the group is the following:

- Giovanni Scotti:
- Marco Trabucchi:

NOTE: indicated hours include the time spent in group work.

Bibliography

[1] AA 2016/2017 Software Engineering 2 - *Requirements Analysis and Specification Document* - Giovanni Scotti, Marco Trabucchi

[2] AA 2016/2017 Software Engineering 2 - *Design Document* - Giovanni Scotti, Marco Trabucchi

[3] AA 2016/2017 Software Engineering 2 - *Project goal, schedule and rules*

[4] SpinGrid Project - *Integration Test Plan*