

Version: 1.0.0

Release date: 16-12-2016

Riccardo Cattaneo 873647

Fabio Chiusano 874294

PowerEnJoy, ITPD

Table of Contents

[1 Introduction 3](#_Toc469679469)

[1.1 Revision History 3](#_Toc469679470)

[1.2 Purose and Scope 3](#_Toc469679471)

[1.3 List of Definitions and Abbreviations 3](#_Toc469679472)

[1.4 List of Reference Documents 3](#_Toc469679473)

[2 Integration Strategy 3](#_Toc469679474)

[2.1 Entry Criteria 3](#_Toc469679475)

[2.2 Elements to be integrated 3](#_Toc469679476)

[2.3 Integration Testing Strategy 3](#_Toc469679477)

[2.4 Sequence of Component – Function Integration 3](#_Toc469679478)

[2.4.1 Software Integration Sequence 3](#_Toc469679479)

[2.4.2 Subsystem Integration Sequence 3](#_Toc469679480)

[3 Individual Steps and Test Description 3](#_Toc469679481)

[4 Tools and Test Equipment Required 3](#_Toc469679482)

[5 Program Stubs and Test Data Required 3](#_Toc469679483)

[6 Effort Spent 3](#_Toc469679484)

# Introduction

## Revision History

### Version 1.0

The first release, it is published online before the deadline.

## Purpose and Scope

This Integration Test Plan Document aims at pointing out how to accomplish integration tests. Developers, testers and, in general, all the people involved in the development of the PowerEnjoy System should read this document before starting testing of the integration of components.

This document needs to explain to the development team what to test, in which sequence, which tools are needed for testing, and which stubs/drivers/oracles need to be developed.

## List of Definitions and Abbreviations

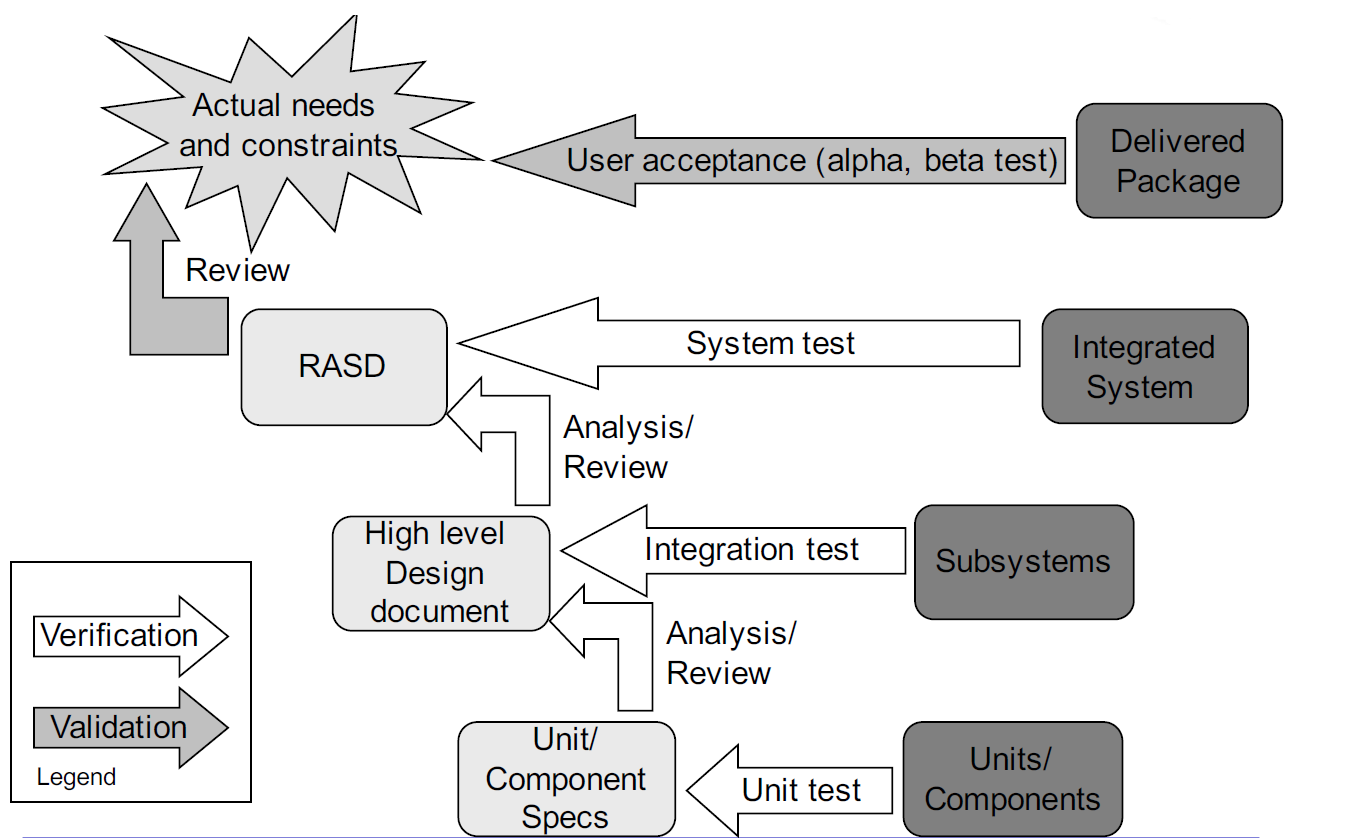
* BL: Business Logic
* ITPD: Integration Test Plan Document
* DD: Design document;
* ER: Entity-Relationship diagram;
* RASD: Requirements Analysis and Specification Document;
* JSE: Java Serial Edition;
* JEB: Java Enterprise Bean;
* REST: REpresantional State Transfer;
* RESTful service: REST compliant service;
* UX: User eXperience;
* JEE: Java Enterprise Edition;
* JAX-RS: Java API for RESTful web Services;
* JBOSS: JEE open source application server;
* JPA: Java Persistence API;
* Cordova: mobile cross-platform development framework;
* PhoneGap: mobile cross-platform development framework that works over Cordova;
* JSP: Java Server Pages.

## List of Reference Documents

* Specification Document: Assignments AA 2016-2017.pdf;
* RASD v1.1 Document;
* Design Document v1.0;
* Example Design Documents from previous years;

# Integration Strategy

## Entry Criteria



We are to design and plan the Integration Test, that aims to verify that software component work with each other and cooperate in the right and expected way.

Hence, it is supposed that each component works well individually and this can be formally proved with Unit Tests.

We assume that Low-level code is already tested, functions of every component are covered with unit tests, with mainly a white-box approach.

## Elements to be integrated

Starting from a high-level view, it is necessary to integrate and test software tiers in the main server: clients with the web tier, web tier with the business logic tier, and this one with the persistence manager and the database. It is also necessary to test interaction between external handler and corresponding controller.

In a more low-level view, it is necessary to integrate every controller with related ones for example the Search Controller with the Map Controller.

It is also necessary to test the interaction between controllers developed in the car software, and between cars and server.

Since the client side is light, it is only necessary to test the interaction with clients and server.

A more detailed list of components to be integrated and tested is present further in the document.

## Integration Testing Strategy

For testing, we choose the bottom-up approach. Since there was not an old system, the project will be built up starting from the ground up.

By choosing bottom-up approach, it is possible to test integration of components as they are ready, with no further delay.

## Sequence of Component – Function Integration

### Software Integration Sequence

Components have to start to be integrated starting from low-level ones.

This process brings to different higher-level and integrated sub-systems.

Every integration test in each section can be performed in parallel with the others.

We suppose that the communication with external handlers, namely between search-on-a-map handler and the Map Controller, and between Payment Handler and Payment Controller, has been already performed while testing single components.

#### Business Logic Components Integration

Thanks to the bottom-up approach, integration testing can be parallelized.

This is a set of integration test that can be performed in parallel, hence the order is not mandatory. They can be carried out as components are finished to be developed and unit tested.

#### Car Components Integration

Software components that run on-board the car are centred on the main controller, called Car Controller. In order to test integration, the Car Controller has to be finished. When the other controllers are finished, they can be integrated and tested with the main one.

### Subsystem Integration Sequence

After having integrated and tested components in the main server and in the car, it is possible to move on higher-level software modules. It is possible to test the integration between tiers on the main server, client and server, server and database, and between car and server.

After doing this, it will be possible to move to the next phase and test the whole system.

# Individual Steps and Test Description

With “possibly”, we mean that a stub must be developed if the related controller is not already finished and usable.

## I1 – Integration Test 1

|  |  |
| --- | --- |
| Test Items | Search Controller – Map Controller |
| Type of Tests | • the Search Controller (SC) can retrieve correctly a map from the Map Controller (MC)  • SC and MC uses same address and coordinates formats  • the SC can apply cars over the map in the right place  • map and address errors are handled correctly  • SC returns to the caller a right formatted map |
| Environmental Needs | Router Driver, possibly Car Controller Stub, Map Handler Stub |

## I2 – Integration Test 2

|  |  |
| --- | --- |
| Test Items | Car Controller – Search Controller |
| Type of Tests | • the Search Controller (SC) can retrieve correctly information of a car (position, battery level, isPluggled, status…) from the Car Controller (CC)  • SC and CC uses same address and coordinates formats  • the SC can find all and only the cars that are available in a given address within a certain range  • car status and address errors are handled correctly |
| Environmental Needs | Router Driver, possibly Map Controller Stub, Map Handler Stub |

## I3 – Integration Test 3

|  |  |
| --- | --- |
| Test Items | User Controller – Login Controller |
| Type of Tests | • the Login Controller (LC) can retrieve correctly user’s information from the User Controller (UC)  • LC and UC uses same data formats  • login errors are handled correctly |
| Environmental Needs | Router Driver |

## I4 – Integration Test 4

|  |  |
| --- | --- |
| Test Items | User Controller – Signup Controller |
| Type of Tests | • the Signup Controller (SC) can retrieve correctly users information from the User Controller (UC)  • SC correctly identifies if a user is already registered to the system  • SC and UC uses same data formats  • signup errors are handled correctly |
| Environmental Needs | Router Driver |

## I5 – Integration Test 5

|  |  |
| --- | --- |
| Test Items | Reservation Controller – Car Controller |
| Type of Tests | • the Reservation Controller (RC) can retrieve correctly a car information from the Car Controller (CC)  • RC and CC uses same data formats  • RC can correctly set the car as reserved  • RC can correctly set and hide the reservation code on the reserved car  • RC and CC can correctly manage the lock and unlock of the car  • reservation and car errors are handled correctly |
| Environmental Needs | Router Driver, Car Handler stub |

## I6 – Integration Test 6

|  |  |
| --- | --- |
| Test Items | Payment Controller – Reservation Controller |
| Type of Tests | • the Reservation Controller (RC) can correctly make a payment via the Payment Controller (PC)  • RC and PC uses same data formats  • PC can provide in an acceptable time the result of the payment  • payment errors are handled correctly |
| Environmental Needs | Router Driver, possibly Payment Handler Stub |

## I7 – Integration Test 7

|  |  |
| --- | --- |
| Test Items | Car Controller – GPS Controller |
| Type of Tests | • the Car Controller (CC) can retrieve correctly actual location information from the GPS Controller (GC)  • CC and GC uses same coordinates formats  • GC returns to the caller a right formatted coordinate  • GPS position errors are handled correctly |
| Environmental Needs | GPS System Stub |

## I8 – Integration Test 8

|  |  |
| --- | --- |
| Test Items | Car Controller – Seat Sensor Controller |
| Type of Tests | • the Car Controller (CC) can retrieve correctly seat information from the Seat Sensor Controller (SC)  • CC can understand how many passengers there are inside the car  • CC and SC uses same data formats  • sensor errors are handled correctly |
| Environmental Needs | Seat Sensor System Stub |

## I9 – Integration Test 9

|  |  |
| --- | --- |
| Test Items | Car Controller – Dashboard Controller |
| Type of Tests | • the Car Controller (CC) can send and show correctly information to the Dashboard Controller (DC)  • CC can show and hide the reservation code  • CC and DC uses same data formats  • dashboard errors are handled correctly |
| Environmental Needs | Main Server BL Driver, Dashboard System Stub |

## I10 – Integration Test 10

|  |  |
| --- | --- |
| Test Items | Car Controller – Taximeter Controller |
| Type of Tests | • the Car Controller (CC) can retrieve correctly taximeter information from the Taximeter Controller (TC)  • CC can start and stop the taximeter  • Calculated fees are proportional to the time elapsed  • taximeter errors are handled correctly |
| Environmental Needs | Taximeter System Stub |

## I11 – Integration Test 11

|  |  |
| --- | --- |
| Test Items | Car Controller – Car Info Controller |
| Type of Tests | • the Car Controller (CC) can retrieve correctly car status information from the Car Info Controller (IC)  • CC can lack doors when is needed  • CC and IC uses same data formats  • sensors and locking errors are handled correctly |
| Environmental Needs | Car System Stub |

## I12 – Integration Test 12

|  |  |
| --- | --- |
| Test Items | Car – Business Logic Controllers |
| Type of Tests | • Controllers, located in the main server, can retrieve correctly information about the status of the car, its position, passengers on it, and others.  • Controllers and car uses same data formats  • The main server can correctly display information on the car  • The main server can understand when the ride is finished, and then perform consequent task  • the communication between server and car satisfy some performance constraints |
| Environmental Needs | Router Driver |

## I13 – Integration Test 13

|  |  |
| --- | --- |
| Test Items | Business Logic Controllers – Persistence Manager |
| Type of Tests | • Controllers can retrieve correctly persistent information through the Persistence Manager (PM)  • Controllers and PM have a compatible data format  • Controllers can save information correctly  • not compatible information (such as out of bound values) are managed correctly |
| Environmental Needs | Router Driver |

## I14 – Integration Test 14

|  |  |
| --- | --- |
| Test Items | Business Logic Controllers – Web Tier |
| Type of Tests | • All possible client request are correctly understood  • Controllers are called in the right order  • Controller answers are consistent and are correctly managed by the web tear  • Web tear answers are well formatted  • possible data error are handled correctly |
| Environmental Needs | Client Driver |

## I15 – Integration Test 15

|  |  |
| --- | --- |
| Test Items | Web App – Web Tier |
| Type of Tests | • The web tier, located on the main server, can manage all possible HTTP requests from the Web App  • The web app display correctly the answer received  • communication protocols are correctly managed  • communication errors are handled gracefully  • HTML forms send by the server are consistent syntactically |
| Environmental Needs | User Driver |

## I16 – Integration Test 16

|  |  |
| --- | --- |
| Test Items | Persistence Manager - Database |
| Type of Tests | • SQL Queries are correctly formed and accepted by the DBMS  • Answers from the database are consistent  • The persistence manager keeps data updated and consistent with the Database  • Data errors are handled correctly |
| Environmental Needs | Possibly Business Logic Driver |

# Tools and Test Equipment Required

# Program Stubs and Test Data Required

# Effort Spent