

**Software Engineering 2: “PowerEnJoy”**

**Integration Test Plan (V. 1.0)**

Authors:

Sergio CAPRARA

Soheil GHANBARI

Erica TINTI

Milan, Italy

11/01/2017

**Table of contents**

[1. Introduction 3](#_Toc470825787)

[1.1 Revision History 3](#_Toc470825788)

[1.2 Purpose and Scope 3](#_Toc470825789)

[1.3 Definitions and Abbreviations 3](#_Toc470825790)

[1.4 Reference Documents 4](#_Toc470825791)

[2. Integration Strategy 5](#_Toc470825792)

[2.1 Entry Criteria 5](#_Toc470825793)

[2.2 Elements to be Integrated 5](#_Toc470825794)

[2.3 Integration Testing Strategy 5](#_Toc470825795)

[2.3.1 Central Application 5](#_Toc470825796)

[2.3.2 Persistence 6](#_Toc470825797)

[2.4 Sequence of Component/Function Integration 6](#_Toc470825798)

[2.4.1 Software Integration Sequence 6](#_Toc470825799)

[2.4.2 Subsystem Integration Sequence 6](#_Toc470825800)

[3. Individual Steps and Test Description 7](#_Toc470825801)

[3.1 Sample 7](#_Toc470825802)

[4. Tools and Test Equipment Required 8](#_Toc470825803)

[4.1 Sample 8](#_Toc470825804)

[4.2 Sample 8](#_Toc470825805)

[5. Program Stubs and Test Data Required 9](#_Toc470825806)

[6. Effort Spent 10](#_Toc470825807)

# Introduction

## Revision History

In this document we are providing.

## Purpose and Scope

In this document, we are providing details on how the components described in the Design Document will be tested. To ensure that the interaction between them will give the expected results, we are choosing the method to follow and we are keeping in mind that Unit Test will be done before starting the Integration Test phase.

In the following chapters you will find detailed descriptions of the tests and the name of the tools to be used.

## Definitions and Abbreviations

* **User:** the person registered to the system and allowed to access to its functions.
* **Operator:** a person with technical skills, that fixes car issues.
* **App:** short term used to define a mobile application.
* **Power Plug:** a column with one or more electricity socket where it is possible to charge the car.
* **Safe Area** (or Parking Area): a parking area with parking shared with all the other divers and not especially reserved to PowerEnjoy.
* **Special Parking Area** (or Power Station): a parking area reserved exclusively to PowerEnjoy cars where, for each parking space there is a Power Plug where it is possible to charge a car.
* **Car:** PowerEnjoy car.
* **Reservation:** the relation between a user and a car, that allows the user to start using the car. The reservation guarantees that no one else can reserve and use the reserved car till the end of the rental.
* **DB:** database, the collection of system data.
* **GUI:** Graphic User Interface, the interface that allows the user to interact with the system.
* **RASD:** Requirements Analysis and Specifications Document.
* **DAO:** Data Access Object.
* **DTO:** Data Transfer Object.
* **MVC:** Model-View-Controller, the pattern used for the development.
* **UX:** user experience.
* **GEB:** Green e-Box.

## Reference Documents

The documents used as a reference to provide the design document are:

* Assignments AA 2016-2017.pdf
* Sample Design Deliverable Discussed on Nov. 2.pdf
* IEEE Standard for IT – System Design – Software Design Description
* Structure of the design document
* Paper on the green move project.pdf
* Second paper on the green move project.pdf

# Integration Strategy

## Entry Criteria

Before starting the Integration testing phase, it’s necessary that the Integration Test Plan document has been completed, in such a way that it is clear how to procede with the integration test. About development and unit testing phase, we have to ensure that classes and methods involved in the integration test of the components that we’re approaching to start have been completely developed and unit tested successfully. In this way we can assure that results produced are meaningful and in the meanwhile we can accelerate the process without waiting that each class and method of the component in testing are completed, but at least the minimum amount of functionality to allow the integration work.

## Elements to be Integrated



Our system is composed by many components, that we can distinguish in two level of granularity.

The higher level of components of our system that needs to be integrated are:

* User Application;
* Operator Application;
* Central Application, containing the main system logic;
* Model
* Database.

These higher level components are composed by some lower level components that needs to be integrated too. More in detail, the component Central Application is composed by Authentication, WebService, Maintenance Controller, Reservation Controller, Calculation Controller. The Model is composed of some DAO components and some Pojo components. The last doesn’t need to be integrated. On DAO Components we won’t perform unit Test. About Data components we have the subcomponents DataService and Database. About the mobile application components we will integrate only the components UserController and OperatorController.

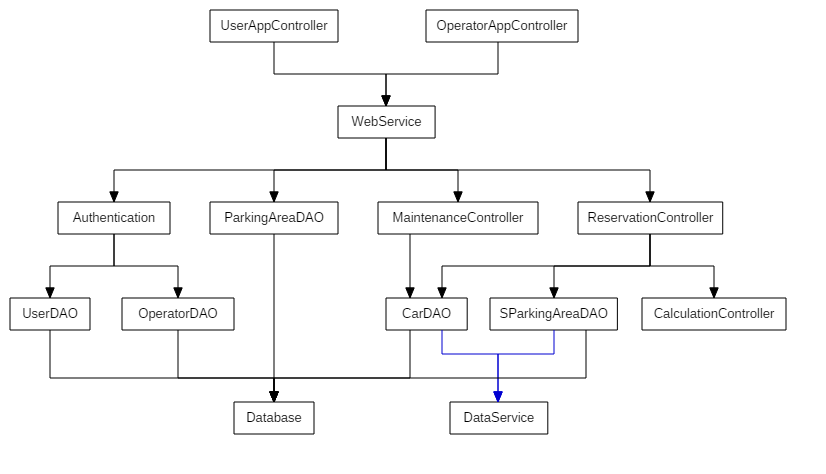
Following the concept “called by” for composing the components hierarchy, we defined a components hierarchy where at the base of the hierarchy we have data components, then on a higher level we have dao components, then the controllers and finally the mobile application, as shown in fig [].The main application interacts with the database to get all information concerning users, operators, cars, parking areas and special parking areas. Most of the interactions concern the request or the update of car information, such as its position before reserving it or after parking it.

## Integration Testing Strategy

For testing the integration among components we will procede following a bottom-up approach that means that, with reference to the component hierarchy defined in paragraph 2.2, we will start from the bottom of the hierarchy and moving to the top on each step. Bottom up approach include the use of drivers for whose components that are ready to be tested but whom component directly above is not ready. The reason why we apply this approach is based on the fact that it guarantees a better result while testing. In fact, we think that the use of drivers, when needed, would be better than working with stubs. Also, this will prevent long working time for making stubs.

As already discussed, the software has many critical points, but we guess that the bottom-up strategy will be a safer method and software faults will be easily found and corrected.

## Sequence of Component/Function Integration



In the following table we provide the list of integration tests required.

|  |  |  |
| --- | --- | --- |
| **ID** | **Integration Test** | **Paragraph** |
| **I1** | UserDAO → Database | 3.1 |
| **I2** | OperatorDAO → Database | 3.2 |
| **I3** | ParkingAreaDAO → Database | 3.3 |
| **I4** | CarDAO → Database  CarDAO → DataService | 3.4 |
| **I5** | SParkingAreaDAO → Database SParkingAreaDAO → DataService | 3.5 |
| **I6** | Authentication → UserDAO  Authentication → OperatorDAO | 3.6 |
| **I7** | MaintenanceController → CarDAO | 3.7 |
| **I8** | ReservationController → CarDAO  ReservationController → SParkingAreaDAO  ReservationController → CalculationController | 3.8 |
| **I9** | WebService → Authentication  WebService → ParkingAreaDAO  WebService → MaintenanceController  WebService → ReservationController | 3.9 |
| **I10** | UserAppController → WebService | 3.10 |
| **I11** | OperatorAppController → WebService | 3.11 |

# Individual Steps and Test Description

## Integration test case I1

|  |  |
| --- | --- |
| **Test Case Identifier** | I1T1 |
| **Test Item(s)** | UserDAO → Database |
| **Input Specification** | User |
| **Output Specification** | The database correctly performs the required action. |
| **Purpose** | Test select, update, insert and delete operations required from the DAO on the database. |
| **Dependencies** | N/A |

## Integration test case I2

|  |  |
| --- | --- |
| **Test Case Identifier** | I2T1 |
| **Test Item(s)** | OperatorDAO → Database |
| **Input Specification** | Operator |
| **Output Specification** | The database correctly performs the required action. |
| **Purpose** | Test select, update, insert and delete operations required from the DAO on the database. |
| **Dependencies** | N/A |

## Integration test case I3

|  |  |
| --- | --- |
| **Test Case Identifier** | I3T1 |
| **Test Item(s)** | ParkingAreaDAO → Database |
| **Input Specification** | ParkingArea |
| **Output Specification** | The database correctly performs the required action. |
| **Purpose** | Test select, update, insert and delete operations required from the DAO on the database. |
| **Dependencies** | N/A |

## Integration test case I4

|  |  |
| --- | --- |
| **Test Case Identifier** | I4T1 |
| **Test Item(s)** | CarDAO → Database |
| **Input Specification** | Car |
| **Output Specification** | The database correctly performs the required action. |
| **Purpose** | Test select, update, insert and delete operations required from the DAO on the database. |
| **Dependencies** | N/A |

|  |  |
| --- | --- |
| **Test Case Identifier** | I4T2 |
| **Test Item(s)** | CarDAO → DataService |
| **Input Specification** | Car |
| **Output Specification** | The information of the required car is provided. |
| **Purpose** | Verify that the DataService can correctly get information on the required car. |
| **Dependencies** | N/A |

## Integration test case I5

|  |  |
| --- | --- |
| **Test Case Identifier** | I5T1 |
| **Test Item(s)** | SParkingAreaDAO → Database |
| **Input Specification** | SpecialParkingArea |
| **Output Specification** | The database correctly performs the required action. |
| **Purpose** | Test select, update, insert and delete operations required from the DAO on the database. |
| **Dependencies** | N/A |

|  |  |
| --- | --- |
| **Test Case Identifier** | I5T2 |
| **Test Item(s)** | SParkingAreaDAO → DataService |
| **Input Specification** | Request for the reservation of a power plug of a certain special parking area. |
| **Output Specification** | The DataService provides the correct response and reserves the designated power plug for the specified power station. |
| **Purpose** | Test the correct reservation of the power plug in the special parking area sent as an input. |
| **Dependencies** | N/A |

## Integration test case I6

|  |  |
| --- | --- |
| **Test Case Identifier** | I6T1 |
| **Test Item(s)** | Authentication → UserDAO |
| **Input Specification** | User |
| **Output Specification** | The DAO correctly verifies information entered for the authentication. |
| **Purpose** | Ensure that the request from the Authentication controller to the DAO receives the right feedback. |
| **Dependencies** | Test Case I1T1 succeeded. |

|  |  |
| --- | --- |
| **Test Case Identifier** | I6T2 |
| **Test Item(s)** | Authentication → OperatorDAO |
| **Input Specification** | Operator |
| **Output Specification** | The DAO correctly verifies information entered for the authentication. |
| **Purpose** | Ensure that the request from the Authentication controller to the DAO receives the right feedback. |
| **Dependencies** | I2T1 succeeded. |

## Integration test case I7

|  |  |
| --- | --- |
| **Test Case Identifier** | I7T1 |
| **Test Item(s)** | MaintenanceController → CarDAO |
| **Input Specification** | Car |
| **Output Specification** | The DAO handles all the incoming requests from the MaintenanceController and can provide the correct answer. |
| **Purpose** | Verify the functions that:   * get the list of cars that need maintenance, * set the status of a car under maintenance, * perform the end of the maintenance request. |
| **Dependencies** | Test Case I4T1 succeeded. |

## Integration test case I8

|  |  |
| --- | --- |
| **Test Case Identifier** | I8T1 |
| **Test Item(s)** | ReservationController → CarDAO |
| **Input Specification** | Car |
| **Output Specification** | The DAO handles all the incoming requests from the ReservationController and can provide the correct answer. |
| **Purpose** | Verify the functions that:   * get the list of the available cars, * set the status of a car in use, * set the status of a car as available after release. |
| **Dependencies** | I4T1 succeeded. |

|  |  |
| --- | --- |
| **Test Case Identifier** | I8T2 |
| **Test Item(s)** | ReservationController → SParkingAreaDAO |
| **Input Specification** | SpecialParking, PowerPlug |
| **Output Specification** | The DAO handles all the incoming requests from the ReservationController and can provide the correct answer. |
| **Purpose** | Verify the functions that:   * get the list of the available power plugs in the special parking areas, * set the status of a power plug and confirm the reservation. |
| **Dependencies** | I5T1 succeeded. |

|  |  |
| --- | --- |
| **Test Case Identifier** | I8T3 |
| **Test Item(s)** | ReservationController → CalculationController |
| **Input Specification** | ReservationInfo |
| **Output Specification** | The CalculationController can provide the expected response to the incoming request. |
| **Purpose** | Verify that the result provided by the CalculationController is correct for the entered parameters. |
| **Dependencies** | N/A |

## Integration test case I9

|  |  |
| --- | --- |
| **Test Case Identifier** | I9T1 |
| **Test Item(s)** | WebService → Authentication |
| **Input Specification** | User or Operator, Request |
| **Output Specification** | The WebService sends the correct request to the Authentication controller and gets the expected response. |
| **Purpose** | Verify that the Authentication controller can correctly verify the entered authentication information for both the user and the operator.  Ensure that the controller can also correctly perform the registration of a new user (new operator not allowed). |
| **Dependencies** | I6T1 and I6T2 succeeded. |

|  |  |
| --- | --- |
| **Test Case Identifier** | I9T2 |
| **Test Item(s)** | WebService → ParkingAreaDAO |
| **Input Specification** | None |
| **Output Specification** | The DAO provides the list of the parking areas. |
| **Purpose** | Test the initialization of the parking areas. The DAO should get the correct information on the database and should return the list of all the parking areas. |
| **Dependencies** | I3T1 succeeded. |

|  |  |
| --- | --- |
| **Test Case Identifier** | I9T3 |
| **Test Item(s)** | WebService → MaintenanceController |
| **Input Specification** | OperatorRequest |
| **Output Specification** | The MaintenanceController correctly interprets the request made by the WebService. |
| **Purpose** | Test the functions through which the operator can:   * take in charge a car for maintenance, * set a car as repaired and ready to use. |
| **Dependencies** | Test Case I7T1 succeeded. |

|  |  |
| --- | --- |
| **Test Case Identifier** | I9T4 |
| **Test Item(s)** | WebService → ReservationController |
| **Input Specification** | UserRequest |
| **Output Specification** | The ReservationController correctly interprets the request made by the WebService. |
| **Purpose** | Test the functions through which the user can:   * request for a reservation, * request a power plug, * release a car. |
| **Dependencies** | I8T1, I8T2, I8T3 succeeded. |

## Integration test case I10

|  |  |
| --- | --- |
| **Test Case Identifier** | I10T1 |
| **Test Item(s)** | UserAppController → WebService |
| **Input Specification** | User, Request |
| **Output Specification** | The WebService dispatches correctly the Request and provides the expected result. |
| **Purpose** | Verify that the Controller can send requests to the WebService and can get the correct response, for the functions to:   * register a new user, * authenticate an already registered user, * reserve a car, * end the use of a car. |
| **Dependencies** | I9T1 and I9T4 succeeded. |

## Integration test case I11

|  |  |
| --- | --- |
| **Test Case Identifier** | I11T1 |
| **Test Item(s)** | OperatorAppController → WebService |
| **Input Specification** | Operator, Request |
| **Output Specification** | The WebService dispatches correctly the Request and provides the expected result. |
| **Purpose** | Verify that the Controller can send requests to the WebService and can get the correct response, for the functions to:   * authenticate an existing operator, * take in charge a car, * end the maintenance of a car. |
| **Dependencies** | I9T1 and I9T3 succeeded. |

# Tools and Test Equipment Required

## Sample

The interface of the mobile applications has already been presented on the RASD, but we wish to add some other screens that we decided to add.

## Sample

The diagram shows how user actions are performed and the sequence of the navigation between the screens.

# Program Stubs and Test Data Required

## Program Stubs and Drivers

For the integration testing of our system we decided to adopt a bottom-up approach. Because of this, we won’t need any program stub, but we will only use the drivers listed here:

* **CarDAO Driver**, **SParkingAreaDAO Driver**: these two modules will invoke the methods exposed by the **CarDAO** and **SParkingAreaDAO** components, for the interaction with the **DataService**. As a recall, the DataService provides a way to make possible the interaction between our system and the system installed on cars and power plugs.
* **Authentication Driver**: the driver module should call the methods of the **Authentication** component to test its interaction with the **UserDAO** and the **OperatorDAO**.
* **MaintenanceController Driver**: this module will invoke the methods exposed by the **MaintenanceController** component to test its interaction with the **CarDAO** component.
* **ReservationController Driver**: this driver will call the methods used by the **ReservationController** component for the interaction with the **CarDAO**, **SParkingAreaDAO** and **CalculationController** components.
* **WebService Driver**: the module should invoke all the methods defined in the **WebService** for the interaction with the **Authentication**, **ParkingAreaDAO**, **MaintenanceController** and **ReservationController** components.
* **UserAppController Driver**, **OperatorAppController Driver**: the drivers will invoke the methods of the **UserAppController** and **OperatorAppController** for their interaction with the **WebService** component.

Drivers are not meant to be defined for the **UserDAO**, **OperatorDAO**, **ParkingAreaDAO**, **CarDAO**, **SParkingAreaDAO** when testing their interaction with the Database,

## Test Data

The Database needs to be filled with meaningful data in order to run tests and tools should be used to avoid wasting time writing one record at a time for all the tables.

**IBM DB2 Test Database Generator** is a tool provided by IBM that can be used for generating data. It allows the definition of:

* the structure of the table,
* constraints on how the data should be generated,
* the output format (SQL, CSV, XML).

Using this tool will be helpful as it will accelerate the testing phase.

Integration tests should also verify the responses of the system in specific cases, such as:

* Null parameters in method call,
* invalid login credentials (User or Operator),
* invalid register information (applies to Users),
* User or Operator with an expired driving licence,
* Car parked outside of the valid Parking Area,
* Reservation of a Car that has already been reserved,
* Payment refused.

More detailed information can be found in the description of the Test Cases, provided in chapter 3.

# Effort Spent

For the document, each one of us has worked around 32 hours.