

Luca Scannapieco - 877145

Andrea Pasquali - 808733

Emanuele Torelli - 876210

22/01/2017

PowerEnJoy

Project Plan Document

Table of contents

[1. Introduction 2](#_Toc472280318)

[1.1. Purpose and scope 2](#_Toc472280319)

[1.2. List of definitions and abbreviations 2](#_Toc472280320)

[2. Project size, cost and effort estimation 2](#_Toc472280321)

[2.1. Size estimation: function points 2](#_Toc472280322)

[2.1.1. Internal logic files 2](#_Toc472280323)

[2.1.2. External logic files 3](#_Toc472280324)

[2.1.3. External inputs 3](#_Toc472280325)

[2.1.4. External outputs 4](#_Toc472280326)

[2.1.5. External inquiries 5](#_Toc472280327)

[2.1.6. Overall estimation 5](#_Toc472280328)

[2.2. COCOMO 5](#_Toc472280329)

[3. Tasks and schedule 5](#_Toc472280330)

[4. Resources allocation 6](#_Toc472280331)

[5. Risk management 6](#_Toc472280332)

[6. Other info 6](#_Toc472280333)

[6.1. Sample documents 6](#_Toc472280334)

[6.2. Used tools 6](#_Toc472280335)

[6.3. Hours of work 6](#_Toc472280336)

[6.4. Changelog 6](#_Toc472280337)

# 1. Introduction

## 1.1. Purpose and scope

## 1.2. List of definitions and abbreviations

* FP: function points
* ILF: internal logic ﬁle
* ELF: external logic ﬁle
* EI: external input
* EO: external output
* EQ: external inquiries

# 2. Project size, cost and effort estimation

## 2.1. Size estimation: function points

The function points approach has the aim to provide an estimation of the size of the project, considering as parameters the functionalities that the system has to accomplish and the complexity of the operations.

To evaluate the size of PowerEnJoy system, we refer to the statistical data provided from the analysis of previous projects, which is reported in the following table.

|  |  |  |  |
| --- | --- | --- | --- |
| *Function types* | *Complexity weight* | | |
| *Low* | *Average* | *High* |
| Internal logic files (ILF) | 7 | 10 | 15 |
| External logic files (ELF) | 5 | 7 | 10 |
| External inputs (EI) | 3 | 4 | 6 |
| External outputs (EO) | 4 | 5 | 7 |
| Eternal inquiries (EQ) | 3 | 4 | 6 |

### 2.1.1. Internal logic files

In this section we are going to find out all the possible ILFs involved in the processes of PowerEnJoy system, and we will indicate for each ILF the estimated complexity according to the table provided before.

* The system has to manage the information about the registered users, which include name, surname, phone number, email, SSN, credit card number and driving licence number, together with a password provided by the system during the registration phase and the always available user’s position. Even if may seem that there are a lot of information about a registered user, they all can be stored in a single table or a flat data structure, so the complexity is low.
* The system has also the necessity to collect information about cars, which include the ID number, the position, the battery level, a Boolean to indicate if it’s in charge, and the state (available, unavailable or out of order).
* A reservation is made of a user and a car, which are non-primitive attributes, and a reservation timer, with is primitive.
* Safe areas consist on a set of positions, and it has a set of power stations
* The position, again, is a “primitive” entity, in the sense of it doesn’t contain any attribute. The complexity is low.
* A bit more complex situation is the management of a ride, because it depends on plenty information, which can be primitive data like duration, number of passenger, total price, battery level at the end of the ride itself, Booleans that indicate the presence of the money saving option, the termination of the ride, if the car is left in charge at the end of the ride, and accidents, as well as non-primitive information like reservation, information about the money saving option, the position and the car.
* The money saving option has a starting position, an ending position and the best power station to leave the car.
* A power station has its own position and a Boolean to indicate the availability.

Here is provided a table as a recap.

|  |  |  |
| --- | --- | --- |
| ILF | Complexity | FPs |
| User | Low | 7 |
| Car | Low | 7 |
| Reservation | Low | 7 |
| Safe area | Low | 7 |
| Position | Low | 7 |
| Ride | Average | 10 |
| Money saving option | Low | 7 |
| Power station | Low | 7 |
| Total | | 59 |

### 2.1.2. External logic files

PowerEnJoy system doesn’t rely on many ELFs, since it doesn’t cooperate with many external services. The only feature it must accomplish is to render maps on the client side applications. Since the interaction is quite complex and there is a big amount of data retrieved, it’s reasonable to think that such ELFs are of high complexity. The following table recaps the complexity of the ELFs:

|  |  |  |
| --- | --- | --- |
| ELF | Complexity | FPs |
| Mapping service on client app | High | 10 |
| Mapping service on car app | High | 10 |
| Mapping service on assistance coordinator program | High | 10 |
| Total | | 30 |

### 2.1.3. External inputs

PowerEnJoy supports many interactions with users among different interfaces. In this section we are going to identify all the main functionalities offered by the system with the corresponding complexity in terms of EIs.

On the client app (mobile app or web app):

* Sign up: this is an operation that involves a quite large number of checks to ensure the validity of the fields, hence the complexity is average and the FPs to take into account are 4.
* Login: this is a simpler operation than the sign up one because involves less checks, therefore the FPs are 3.
* Make a reservation: this is a simple operation because involves only the creation of the object Reservation associated to the corresponding user and car, so it contributes 3 FPs.
* Unlock a car: this operation is quite complex, because it need to check if the user is trying to unlock the car he had reserved, and it has also to check if he is less distant than 5m from the car. The FPs contributed are 6.

On the car app:

* Start a ride: this operation is not complex since it only creates the object Ride associated to the right user, so it yields 3 FPs.
* Finish a ride: this operation involves a flow of information between the server and the car, since the server has to check if the user is attempting to park the car in a safe area (or the sensors detected that an accident occurred). The complexity is average, so the FPs produced are 4.
* Plug the car into a power station: once a user terminates his ride, he has 3 minutes to eventually plug the car into a power station in order to get a discount. This operation is trivial, so it contributes 3 FPs.

On the assistance coordinator program:

* Login: as the counterpart used to authenticate the access of user, this is a simple operation and yields 3 FPs.
* Tag/untag a car as out of order: this is a simple operation which produces 3 FPs.

Here is a table that recaps the paragraph:

|  |  |  |
| --- | --- | --- |
| EI | Complexity | FPs |
| Sign up | Average | 4 |
| Login (user) | Low | 3 |
| Make a reservation | Low | 3 |
| Unlock a car | High | 6 |
| Start a ride | Low | 3 |
| Finish a ride | Average | 4 |
| Plug the car into a power station | Low | 3 |
| Login (assistance coordinator) | Low | 3 |
| Tag/untag a car as out of order | Low | 3 |
| Total | |  |

### 2.1.4. External outputs

As a part of its normal functionalities, the PowerEnJoy system occasionally needs to communicate with the user outside the context of an inquiry. These occasions are:

* Provide a password to the user (end of sign up process): this operation may be seen as a part of the signing up procedure, but due to a significant elaboration of logic files, we decided to split them up and to indicate the giving of the password to the user as an external output procedure. This is not a complex procedure, so it yields 4 FPs.
* See all the information about a ride: during a ride, the car app is supposed to show the necessary information about a ride, like cost and duration, as well as a map with safe areas and power station. Since this information involves a quite massive retrieval of data, the FPs produced are 5.
* Show the final cost of the ride: this is a complex operation since the system has to check a large number of parameters including the car’s battery, the car’s position, the number of passengers during a ride, if the car is left in charge and so on. All of these operations are in a strict sequence due to the priorities that the various discounts hold. Therefore the complexity is very high and the contribute is 7 FPs.
* Apply the 1 Euro fine if the reservation expires: this operation is simple because it only needs to check when a reservation gets out of the allowed time. The contribute is 4 FPs.

Here is a table as a recap:

|  |  |  |
| --- | --- | --- |
| EO | Complexity | FPs |
| Provide a password to the user (end of sign up process) | Low | 4 |
| See all the information about a ride | Average | 5 |
| Show the final cost of the ride | High | 7 |
| Apply the 1 Euro fine if the reservation expires | Low | 4 |
| Total | |  |

### 2.1.5. External inquiries

An inquiry is a data retrieval request performed by a user.

* See available cars (user – current position): the user clicks a button and then he’s allowed to see all the cars in a map, so it is a trivial operation and contributes 3 FPs.
* See available cars (user – given address): almost the same operation mentioned before, with the difference that instead of retrieving the user’s position, the address is given by the user himself. The FPs produced are 3.
* See position and battery of cars (assistance coordinators): it’s essentially the same operation that a user can do, except from the fact that the assistance coordinator can see all the cars, therefore the contribution is again 3 FPs.
* Show battery and position of the reserved car: the system allows the owner of a reservation to see the position and the battery of the reserved car. This operation is simple, it produces 3 FPs.
* Activate the money saving option: this is probably the most complex inquiry, because the system has to display, according to the destination provided by the user and the distribution of the cars within the city, the best power station where the user can leave the car. This operation contributes 6 FPs.

The following table shows a recap:

|  |  |  |
| --- | --- | --- |
| EQ | Complexity | FPs |
| See available cars (user – current position) | Low | 3 |
| See available cars (user – given address) | Low | 3 |
| See position and battery of cars (assistance coordinators) | Low | 3 |
| Show battery and position of the reserved car | Low | 3 |
| Activate the money saving option | High | 6 |
| Total | |  |

### 2.1.6. Overall estimation

The following table provides a summary of the previous analysis:

|  |  |
| --- | --- |
| *Function Type* | *Value* |
| Internal logic files (ILF) |  |
| External logic files (ELF) |  |
| External inputs (EI) |  |
| External outputs (EO) |  |
| Eternal inquiries (EQ) |  |
| *Total* |  |

Thus, we can estimate the total lines of code.

## 2.2. COCOMO

# 3. Tasks and schedule

# 4. Resources allocation

# 5. Risk management

# 6. Other info

## 6.1. Sample documents

* Assignments AA 2016-2017.pdf
* Documents previously provided:
  + PowerEnJoy – RASD.pdf
  + PowerEnJoy – DD.pdf
  + PowerEnJoy – ITPD.pdf
* Sample documents:
  + Project planning example document.pdf
* Course slides:
  + Project Management Basics + Advanced.pdf

## 6.2. Used tools

* Microsoft Word 2016, for the drafting of the ITPD
* Microsoft OneDrive, to allow concurrent editing
* GitHub, to store the project in a repo

## 6.3. Hours of work

For redacting and writing the Project Plan Document we spent approximately 25 hours per person.

## 6.4. Changelog

No changes in the document for the moment.