

Luca Scannapieco - 877145

Andrea Pasquali - 808733

Emanuele Torelli - 876210

22/01/2017

PowerEnJoy

Project Plan Document

Table of contents

[1. Introduction 2](#_Toc472360396)

[1.1. Purpose and scope 2](#_Toc472360397)

[1.2. List of definitions and abbreviations 2](#_Toc472360398)

[2. Size estimation: function points 2](#_Toc472360399)

[2.1. Internal logic files 2](#_Toc472360400)

[2.2. External logic files 3](#_Toc472360401)

[2.3. External inputs 4](#_Toc472360402)

[2.4. External outputs 5](#_Toc472360403)

[2.5. External inquiries 5](#_Toc472360404)

[2.6. Overall estimation 6](#_Toc472360405)

[3. Cost and effort estimation: COCOMO II 7](#_Toc472360406)

[3.1. Post-architecture or Early-design 7](#_Toc472360407)

[3.2 Size parameter estimation 7](#_Toc472360408)

[3.3 Scale factors 7](#_Toc472360409)

[3.3.1. Precedentedness (PREC) 8](#_Toc472360410)

[3.3.2. Development flexibility (FLEX) 9](#_Toc472360411)

[3.3.3 Risk Resolution (RESL) 9](#_Toc472360412)

[3.3.4 Team Cohesion (TEAM) 10](#_Toc472360413)

[3.3.5 Process Maturity (PMAT) 10](#_Toc472360414)

[3.4 Cost drivers 11](#_Toc472360415)

[4. Tasks and schedule 11](#_Toc472360416)

[5. Resources allocation 11](#_Toc472360417)

[6. Risk management 11](#_Toc472360418)

[7. Other info 11](#_Toc472360419)

[7.1. Sample documents 11](#_Toc472360420)

[7.2. Used tools 11](#_Toc472360421)

[7.3. Hours of work 11](#_Toc472360422)

[7.4. Changelog 11](#_Toc472360423)

# 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. 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. 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 and the FPs produced are 7.
* 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), so the complexity is low and the FPs produced are 7.
* A reservation is made of a user and a car, together with a reservation timer, so the complexity is low and the FPs produced are 7
* Safe areas consist on a set of positions, and have a set of power stations, so because of the large amount of data they can contain, the complexity is high and the FPs produced are 15.
* The position consists only of latitude and longitude, so the complexity is low and the FPs produced are 7.
* 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, some 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 other data objects like reservation, information about the money saving option, the position and the car. The complexity is average and the FPs produced are 10.
* The money saving option has a starting position, an ending position and the best power station to leave the car. The complexity is low and the FPs produced are 7.
* A power station has its own position and a Boolean to indicate the availability. The complexity is low and the FPs produced are 7.

Here is provided a table as a recap.

|  |  |  |
| --- | --- | --- |
| ILF | Complexity | FPs |
| User | Low | 7 |
| Car | Low | 7 |
| Reservation | Low | 7 |
| Safe area | High | 15 |
| Position | Low | 7 |
| Ride | Average | 10 |
| Money saving option | Low | 7 |
| Power station | Low | 7 |
| Total | | 67 |

## 2.2. 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 |

## 2.3. 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.
* 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.
* 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 quite simple, since it has to check only a condition when a timer expires, so it contributes 3 FPs.

On the assistance coordinator program:

* Login: as the counterpart used to authenticate the access of users, this is a simple operation and yields 3 FPs.
* Tag/untag a car as out of order: this is a simple operation which only changes an attribute of the selected car, therefore it 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 |
| Activate the money saving option | High | 6 |
| 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 | | 38 |

## 2.4. 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 stations. Since this procedure involves a quite massive retrieval of data, we consider this operation as complex and the FPs produced are 7.
* 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 | High | 7 |
| Show the final cost of the ride | High | 7 |
| Apply the 1 Euro fine if the reservation expires | Low | 4 |
| Total | | 22 |

## 2.5. 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.

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 |
| Total | | 12 |

## 2.6. Overall estimation

The following table provides a summary of the previous analysis:

|  |  |
| --- | --- |
| *Function Type* | *Value* |
| Internal logic files (ILF) | 67 |
| External logic files (ELF) | 30 |
| External inputs (EI) | 38 |
| External outputs (EO) | 22 |
| Eternal inquiries (EQ) | 12 |
| *Total* | 169 |

# 3. Cost and effort estimation: COCOMO II

In this section we will adopt the COCOMO approach in order to get an estimation of effort that is needed to spend to complete the PowerEnjoy project. This method is essentially based on the following basic formula that estimates the effort in Person-Month:

Where:

* A=2.94 is a COCOMO constant.
* Size is the result of the size estimation
* E is given from the following formula:where:  
  B=0.91 is a COCOMO constant  
   is the j-th scale factor (computed in the section 2.2.2)
* is the i-th cost driver (computed in the section 2.2.3)

In essence, we are going to model, as more accurately as possible many aspects of our project and compute, on this model, the value of the PM index.

## 

## 3.1. Post-architecture or Early-design

Even if we are projecting a totally new artefact we are going to carry out our effort estimation following the post-architecture model. This choice is given from the fact that, at this stage of our work, we have already chosen a quite detailed architecture for our system to be. In fact we have already delivered the design document to our customers, so we think we have enough information about the architecture of our system to follow the post-architecture model.

## 

## 3.2 Size parameter estimation

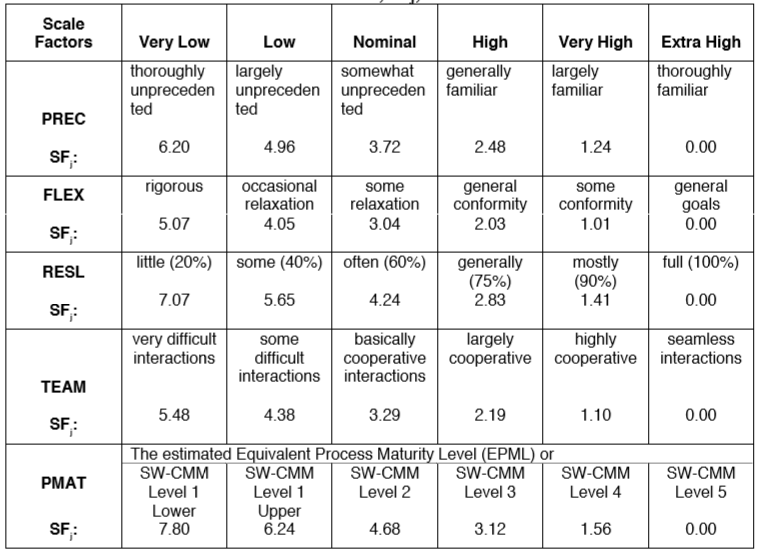
The *Size* parameter is the estimated size of the project, expressed in KSLOC (Kilo-Source Lines Of Code). It can be deducted from the UFP computed in the section 2.1 with the following formula:

i.e. it’s the product of the number of functional points and a parameter that expresses the average number of LOC per FP for a given programming language. Since we are going to build our project with JEE this parameter is (source: Quantitative Software Management):

## 

## 3.3 Scale factors

The following official COCOMO II table shows the value of the five COCOMO scale factors for each rating level (from very low to extra high).



In the following five sections we are going to argue the choice of the rating level for each scale factor. These choices are taken considering, for each scale factor, a criteria (illustrated as tables that be found in the COCOMO model definition manual) that map some project features onto the scale factors.

### 

### 3.3.1. Precedentedness (PREC)

It’s high if the product is similar to several previously developed projects.

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Very low** | **Nominal/High** | **Extra High** |
| Organizational understanding of product | General | Considerable | Thorough |
| Experience in working with related software systems | Moderate | Considerable | Extensive |
| Concurrent development of associated new hardware and operational procedures | Extensive | Moderate | Some |
| Need for innovative data processing architectures, algorithms | Considerable | Some | Minimal |

No particular relationship between developers and costumers leads to a general organizational understanding of the product. For all the members of our group this is the first large scale project development. The rating level is LOW.

### 3.3.2. Development flexibility (FLEX)

It reflects the degree of flexibility in the development process with respect to the external specification and requirements.

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Very low** | **Nominal/High** | **Extra High** |
| Need for software conformance with pre-established requirements | Full | Considerable | Basic |
| Need for software conformance with external interface specifications | Full | Considerable | Basic |

High need of meeting the requirements but just basic needs of conformance with external interfaces.

Chosen rating level: HIGH.

### 

### 3.3.3 Risk Resolution (RESL)

It reﬂects the level of awareness and reactiveness with respect to risks.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Characteristic** | **Very Low** | **Low** | **Nominal** | **High** | **Very High** | **Extra High** |
| Risk Management Plan identifies all critical risk items, establishes milestones for resolving them by PDR or LCA. | None | Little | Some | Generally | Mostly | Fully |
| Schedule, budget, and internal milestones through PDR or LCA compatible with Risk Management Plan. | None | Little | Some | Generally | Mostly | Fully |
| Percent of development schedule devoted to establishing architecture, given general product objectives. | 5 | 10 | 17 | 25 | 33 | 40 |
| Percent of required top software architects available to project. | 20 | 40 | 60 | 80 | 100 | 120 |
| Tool support available for resolving risk items, developing and verifying architectural specs | None | Little | Some | Good | Strong | Full |
| Level of uncertainty in key architecture drivers: mission, user interface, COTS, hardware, technology, performance. | Extreme | Significant | Considerable | Some | Little | Very little |
| Number and criticality of risk items. | >10 Critical | 5-10  Critical | 2-4  Critical | 1 Critical | >5  Non-Critical | <5  Non-Critical |

### 

### 3.3.4 Team Cohesion (TEAM)

It reflects the degree of difficulty in synchronizing the project’s stakeholders: users, customers, developers, maintainers.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Characteristic** | **Very Low** | **Low** | **Nominal** | **High** | **Very High** | **Extra High** |
| Consistency of stakeholder objectives and cultures | Little | Some | Basic | Considerable | Strong | Full |
| Ability, willingness of stakeholders to accommodate other stakeholders’ objectives | Little | Some | Basic | Considerable | Strong | Full |
| Experience of stakeholders in operating as a team | None | Little | Little | Basic | Considerable | Extensive |
| Stakeholder teambuilding to achieve shared vision and commitments | None | Little | Little | Basic | Considerable | Extensive |

The members of our team cooperate very well, but we have to take into account the inevitable divergences with other stakeholders such as users and costumers. The rating level is HIGH.

### 

### 3.3.5 Process Maturity (PMAT)

Refers to a well known method for assessing the maturity of a software organization, CMM, now evolved into CMMI.

|  |  |  |
| --- | --- | --- |
| **PMAT Rating** | **Maturity Level** | **Process Characteristics** |
| **Very low/low** | CMM level 1  Initial | Process is unpredictable, poorly controlled, and reactive |
| **Nominal** | CMM level 2  Managed | Process is characterized for projects and is often reactive |
| **High** | CMM level 3  Defined | Process is characterized for the organization and is proactive |
| **Very high** | CMM level 4  Quantitatively managed | Process is measured and controlled |
| **Extra high** | CMM level 5  Optimizing | Focus is on continuous quantitative improvement |

We think our project is not so advanced to contain proactive processes. It’s often reactive and relies on the knowledge of the three group members and their good interaction. The rating level is NOMINAL.

## 3.4 Cost drivers

# 4. Tasks and schedule

# 5. Resources allocation

# 6. Risk management

In this section there is a list of all the risks that may affect the project and the possible troubles it may encounter.

First of all, we must be sure that with a car sharing service like PowerEnJoy, we are entrusting real cars to people. We can’t ensure that PowerEnJoy users always drive responsively and accidents may happen, so the company has to necessarily stipulate an advantageous insurance policy to cope with steals, accidents, fire, or other damages to cars. Anyway, the company assesses that during a ride, the responsibility of the car is given only to the pending user. In any case the company will necessarily hire some operators to deal with accidents or cars left out of battery around the city.

We also have to consider that PowerEnJoy isn’t the only car sharing system available in Milano, so the society has to be ready to promote the service with consistent strategies of marketing. For the same reason, it must be considered that other companies can constantly improve their services by cutting costs and improve the functionalities of the system, so the society has to be ready to revise some of the policies in order to maintain faith of the clients. As for the PowerEnJoy system development, we believe that a friendly user interface and the facility of usage of the application can help in a very consistent way the spreading of the service among citizens, so the society better hire one or more designers to help the developers to achieve the most appetizing product possible.

Another important consideration is that users can be rude-behaving while using PowerEnJoy cars, therefore, in addition to a normal maintenance of the bodywork and internals of cars, also the devices that run the car app are subject to damages produced by a massive usage from the users (e.g. they can break the touchscreen while tapping on it too violently).

A problem that can occur during the implementation of the system is that engineers have to know how to design and develop software for different platforms that obviously require different programming languages and programming paradigms. In fact, if we think about the client side apps, only to develop the client app we need an iOS programmer with Swift knowledge for Apple devices, an Android programmer with Java knowledge, eventually a Windows programmer with C# knowledge, in addition to a web programmer to develop the web app which requires knowledge of many languages like HTML, CSS, JavaScript and PHP. Since is difficult to find expert programmers who know how to develop for all the possible platforms, the society has the need of recruiting engineers with different backgrounds. Another possible way to reduce communication issues within the team is to look for cross-platform developers, like Cordova or Xamarin programmers, with the benefit that the application of every mobile platform is written once in the same language and run everywhere.

Another issue is related to our dependency on external services and components. PowerEnJoy doesn’t depend on many external services, but for example the mapping service is one of core functionalities of every client app. A change in the terms and conditions of the mapping service, or even just a modiﬁcation of the API itself, could pose serious technical problems. A possible countermeasure is to choose the most reliable and open to third party software mapping service and design the code to be as portable as possible and with a great modularity and independence between components.

Of course, since the system relies on a huge quantity of written code, losing some parts of the source code would be a disaster, but in this case the issue is more easy to tackle, since the staff can rely on versioning control systems and backup services to prevent the loss of any data.

Another possible issue is to integrate the physical sensors and equipment of the cars with the PowerEnJoy system, the staff must be sure that before starting developing software with a certain programming language instead of others, the hardware must be integrated with a full compatibility. Probably the most critical component is the on-board device which runs the car app: the engineers have to choose the most reliable component and the easiest to integrate, obviously before starting developing the car app.

Another important thing is letting the stakeholders have an active role in the development of the project, in the requirement analysis and design phases as well as in the implementation phase. Activities in this direction may include periodical reviews and meetings, demonstrations, discussions on the interface design and so on. We have to be conscious that putting together the requirements and desires of the stakeholders may not be an easy task and that some negotiations are certainly going to be there.

A ﬁnal problem may also emerge from issues with the project scheduling. Even though an initial overall schedule is provided in this document, it can’t obviously take into account all the possible issues that may arise down the road. For this reason, some extra time has been allocated at the expected end of each major activity to allow for adjustments.

# 7. Other info

## 7.1. 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
  + COCOMO II – Model Definition Manual
* Course slides:
  + Project Management Basics + Advanced.pdf

## 7.2. 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

## 7.3. Hours of work

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

## 7.4. Changelog

No changes in the document for the moment.