Politecnico di Milano

Software Engineering 2 Project Work

PowerEnJoy Project: Project Plan Document

Version: 1.0

Autohrs:

Jacopo Fantin (mat. 878723)

Francesco Larghi (mat. 876928)

Professors:

Elisabetta Di Nitto

Luca Mottola

January 22, 2017

Contents

1. Introduction

1.1 Revision History

Version	Date	Author(s)	Summary
1.0	22/01/2017	Francesco Larghi - Jacopo Fantin	First release

1.2 Purpose and Scope

This is the Project Plan Document (PPD) of our PowerEnJoy project system.
The purpose of this document is to give an overall organization and estimation of the work from the documentation to the devolpment and the deployment of the system in order not to waste the available time and the man power. This is useful to define a proper budget, time needed, resource allocation and the schedule of the activities. These informations are absolutely significant and fundamental for the stakeholders.
In the first part we will use Function Points and COCOMO techniques to estimate the expected size of our project in terms of time and lines of code and the cost/effort required for the development.
In the second part we will propose an idea for the schedule of all the project phases and activities.
Then we will propose possible assignments of resources (members of our group) for each identified task. Finally, we will focus on possible risks of the project and general conclusions.

1.3 List of Definitions and Abbreviations

- **PPD:** Project Plan Document.
- **RASD:** Reqirements Analysis Specifications Document.
- **DD:** Design Document.
- **DB:** DataBase.
- **DBMS:** DataBase Management System.

- **Component:** Software element that implements functionalities.
- **FP:** Function Points.
- **ILF:** Internal Logic File.
- **EIF:** External Interface File.
- **EI:** External Inputs.
- **EO:** External Outputs.
- **EQ:** External Inquiries.

1.4 List of Reference Documents

– Our PowerEnjoy Requirements Analysis Specifications Document

– Our PowerEnjoy Design Document

– Our PowerEnjoy Integration Testing Plan Document

– The specification document: Assignments AA'16-'17.pdf

– Examples of Project Plan Document available

2. Project Size, Cost and Effort Estimation

This chapter regards the assessment of the required resourced in order for the project PowerEnJoy to be completed. This estimate concerns size estimation at first, that can be well enough represented through the number of Lines Of Code, and then cost and effort estimation at second, which corresponds to an estimate of how much time the entire development process will take.

2.1 Size Estimation: Function Points

To estabilish an approximated size for the project, the Function Points approach has been employed. This method dates back to 1975 and bases itself on estimation from the services it offers. For each functionality, a number of FPs is to be assigned based on the weight the function is believed to have. The sum of the FPs will provide the Unadjusted Function Points from which to derive the LOC number. According to this algorithm, functions are grouped by function types, namely:

- **Internal Logical Files**, set of data generated and managed by the application itself;
- **External Logical Files**, set of data that the application makes use of but are generated and managed by other applications;
- **External Input**, functionalities that process data coming from the external environment;
- **External Output**, functions that provide data for external applications to be processed;
- **External Inquiry**, operations involving input and output substantially without elaboration of data coming from logical files.

Each function or data element is given a grade of complexity, i.e. low, average or high, looking at the amount of fields that compose it. This done, we assigned an amount of FPs to each element according to the function type it belongs to, as shown in the following table:

Function Type	Low	Avg	High
ILF	7	10	15
EIF	5	7	10
EI	3	4	6
EO	4	5	7
EQ	3	4	6

We now move to considering every single function type of those mentioned above to estimate the number of FP to be assigned to each of them.

2.1.1 Internal Logical Files (ILF)

Our system makes use of several data that are internally stored and provide almost every information the service needs for its purpose. These data can be divided as follows:

User data These pieces of information are necessary for the user attributes and comprise: user ID, password, name, surname, email address, driving license number, credit card number and location.

Car data Store information each car registered in the system has to hold and include: car plate, status, last position, battery level, and whether the car is plugged in or not.

Reservation data These are attributes that indicate date and time of a reservation, its ID, the related user and car involved.

Ride data Give information about the recorded rides, such as status, the distance, the date, the corresponding reservation, the price and potential discounts and fees that could be applied to the final price.

Payment data The Payment entity stores simple information such as state of the payment and amount, beyond the indication of the reservation to which it is connected.

PowerGridStation data These are fields showing information about power grid stations, i.e. the corresponding ID number, the position, and the station status.

SafeArea data In order to know everything about the safe area, we just have to recognize its boundaries through their locations.

All of the fields of these elements can be considered as simple pieces of information, exept for the Reservation and Ride entities, which are more complex since they have to deal with more computations and are linked to other elements' fields. Thus, we are allowed to bestow the Medium complexity on Reservation data and Ride data, and the Low one for each of the others. The total amount of FP for the ILF function type is $10 \cdot 2 + 7 \cdot 5 = 55 FPs$.

2.1.2 External Interface Files (EIF)

The rest of the files the system needs is provided by the external environment, in particular these information are location data coming from Google Maps API. We can assume that these data have in general a medium complex structure, for the total amount of FPs for EIF is $7 \cdot 1 = 7FPs$.

2.1.3 External Inputs (EI)

The main operations that the user has to be able to perform are:

Registration, login, logout: these three are basic functions that don't require much implementation efforts. We can model them with low complexity items.

Reserve a car: making a reservation is not trivial as it involves a User, a Car and a Reservation object itself. It can be weighted as of average complexity.

Cancel a reservation: this is an easier function than the reservation one, so we can confer low complexity on it.

End the ride: it involves a Ride and a Payment object, since it computes the corresponding bill for a specific service employment. Medium complexity.

The total number of FPs for the EI function type is therefore $3 \cdot 3 + 4 + 3 + 4 = 20FPs$.

2.1.4 External Outputs (EO)

PowerEnJoy provides as outputs for the user usage:

Send email: after a reservation a password must be sent to the user via email, who will use the password to unlock the reserved car. Simple operation.

Show map: in this function we intend to include the position of user, cars, and power grid stations, so it counts for three distinct entities of average complexity.

Send receipt: this operation is performed once the ride comes to an end. It delivers a receipt so that the right amount of money (taking into account discounts and additional fees) is subtracted from the user's credit card for the payment. It deals with a bank account, so at least we could consider this as a medium complexity function.

Summing these quantities up, we obtain a number of $4 + 5 \cdot 3 + 5 = 24FPs$.

2.1.5 External Inquiries (EQ)

For the case of interest, we have a limited number of External Inquiries that allow the customer to request information about:

User profile: details on personal information in the user's reserved area.

Reservation details: list of the reservations that have been confirmed by the user.

Both of these operations can be considered as low complexity operations, so that we allocate $3 \cdot 2 = 6FPs$ for the EQ function type.

2.1.6 Overall Estimation

Function Type	Value
ILF	55
EIF	7
EI	20
EO	24
EQ	6
Tot	**112**

This total quantity represents our UFP number.

The Function Points method now allows to compute a low and a high boundary between which lays the estimated number of LOC that the software project will consist of. Based on the previously found total UFPs, these boundaries are given by the formula

$$LOC = AVC \cdot UFP$$

where AVC is a language-dependent multiplication factor. According to the Function Point Language Table, version 5.0. Since we're developing this project on the J2EE platform, this factor equals 15 for the lower bound and 67 for the upper bound for the number of LOC. This way, we can conclude that the approximated number of LOC stays between the two bounds

$$\textit{lower bound } LOC = 15 \cdot 112 = 1680\ LOC$$

$$\textit{upper bound } LOC = 67 \cdot 112 = 7504\ LOC$$

2.2 Cost and Effort Estimation: COCOMO II

2.2.1 Scale Drivers

2.2.2 Cost Drivers

2.2.3 Effort Equation

2.2.4 Schedule Estimation

3. Project Plan Schedule

In this part, as we said in the introduction, we will provide an idea of the possible schedule for the whole project. Of course, the schedule estimation will be more precise the more the project will be completed. This could be a first kind of organization of the general work with main deadlines. In order to mantain the readability of the schedules, we decide to divide it into its main parts:

- **RASD**
- **DD**
- **Development**
- **Deployment**

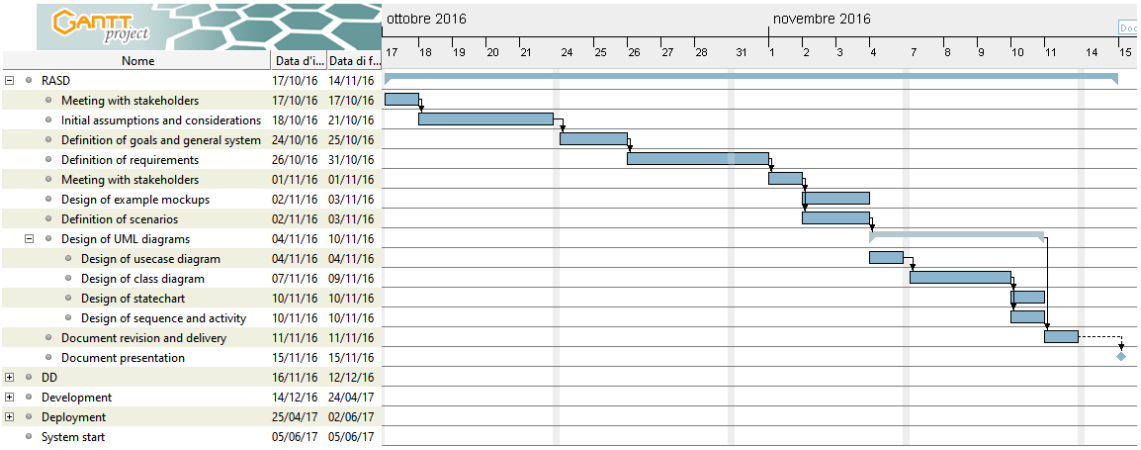

Figure 3.0.1: RASD schedule

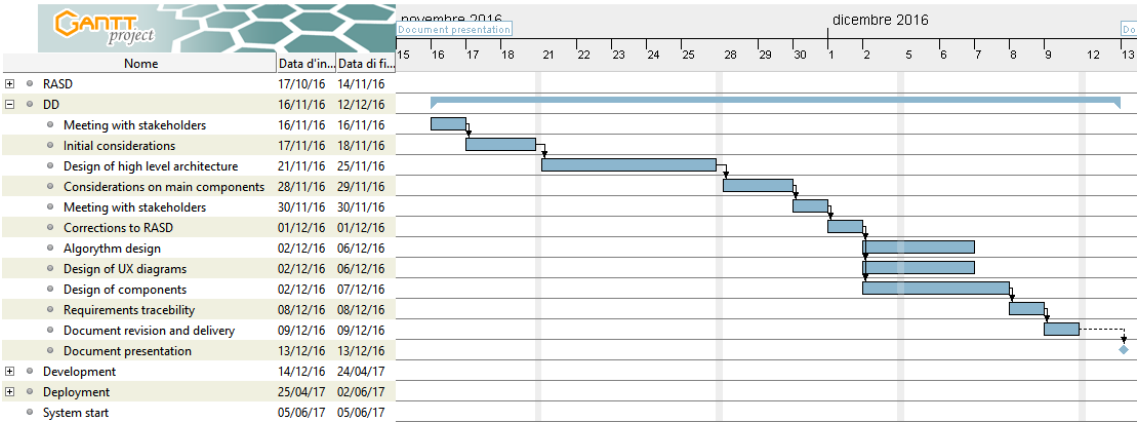

Figure 3.0.2: DD schedule

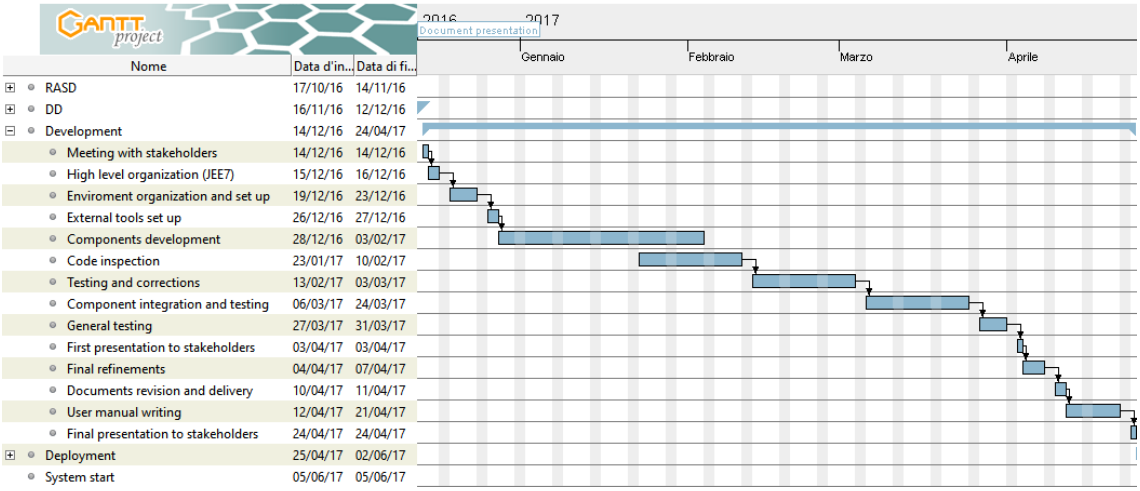

Figure 3.0.3: Development schedule

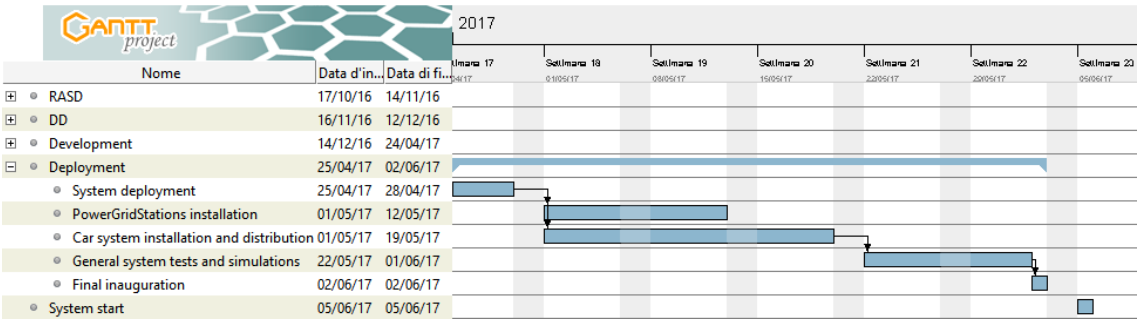

Figure 3.0.4: Deployment and start schedule

4. Resource Allocation

Here we will show the resources assignment to each task identified in the previous chapter. Of course, this is only a simualtion, the development and deployment phase will never start. This is just a possbile way to split the work between the components of our group, even if we think that a project like this should be handled by more than just two persons. In fact, when there are more than two tasks simultaneously the resource allocation is highlighted in blue and this means that that resource is overworking to compensate the lack (just an example to show what happens when there are many simultaneous tasks). In order to mantain th readability again, we split the diagrams for each rescource (the components of our group).

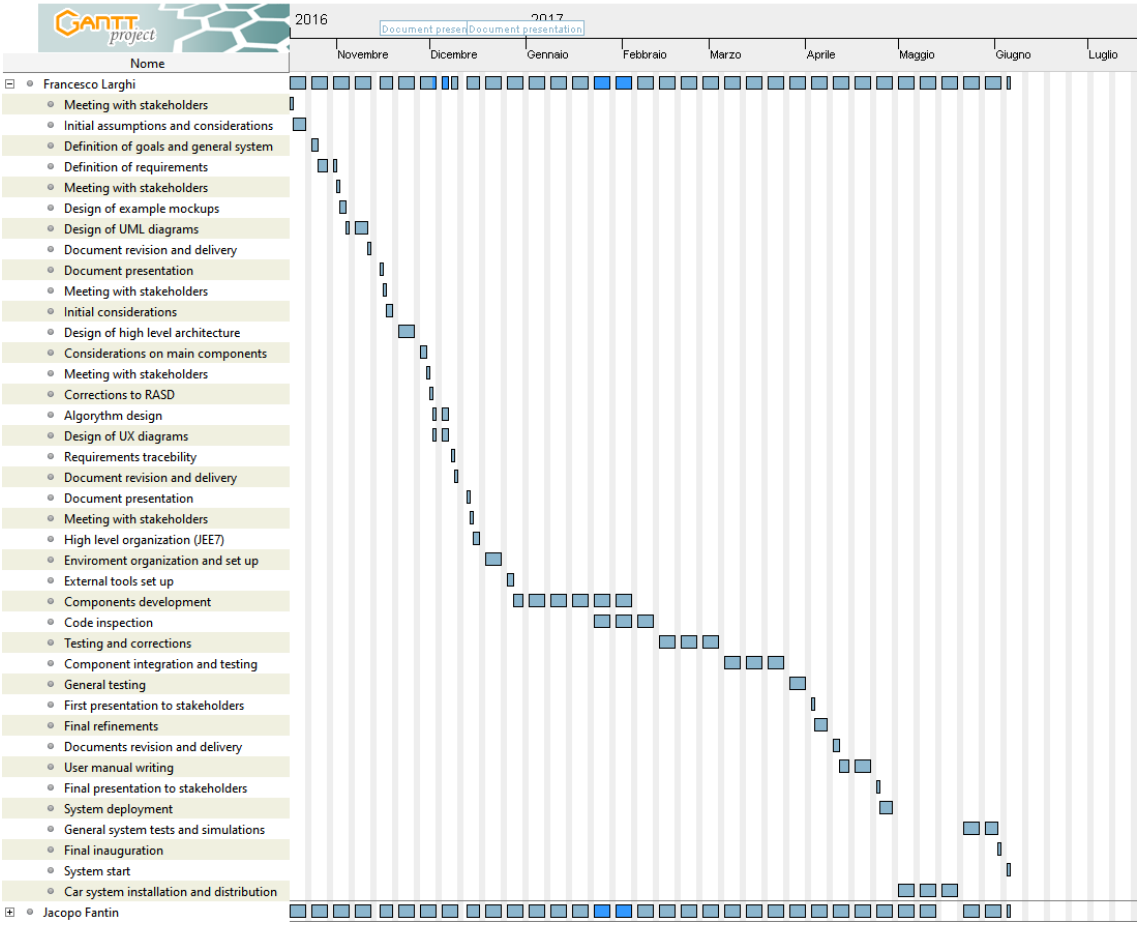

Figure 4.0.1: Rescource allocation for Francesco Larghi

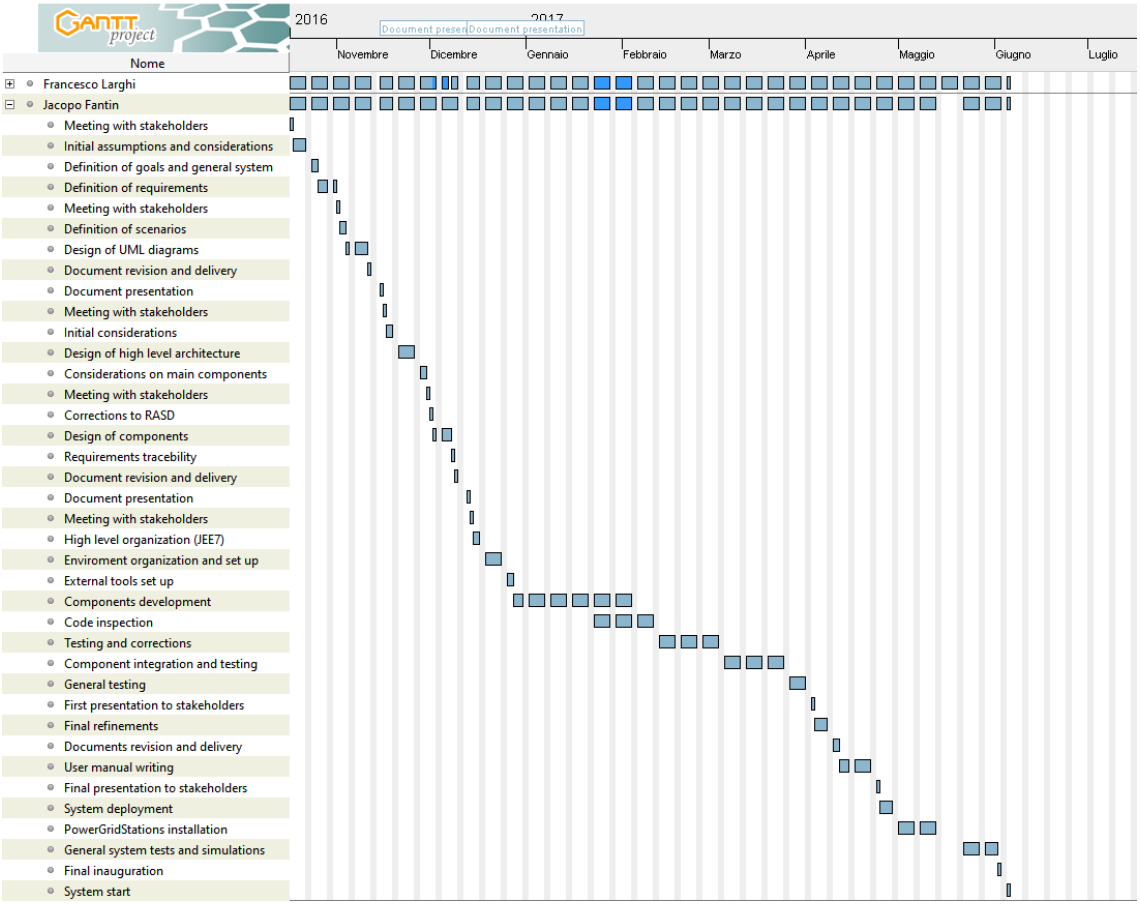

Figure 4.0.2: Rescource allocation for Jacopo Fantin

5. Risk Management

Here we want to define the principle risks that our team could face.
There could be technical problems, or maybe financial or political ones. This last category could cause big problem to the whole system, so it would be a good practice mantaining a good relationship with local government and anticipate possible changes of laws or standards during the project initial phases. National legislation could change too. Traffic laws changes quite often, maybe just in little particulars but these can cause big problems to our system if not predicted before. For example, if the standard for the ID or the payment system or the driving licence changes after the software was already developed it will be a big issue for the team that implemented and tested relative components. These kind of issues could bring a big loss of time and money, and a delay on the project deadlines, in addition to inconveniences to the developers that have worked for nothing.
A way to contain possible problem like these is to get involved the maximum number of people in the development of the project in order to have multiple opinions and mantain the developers updated on every kind of possible changes of the external enviroment.

The same rules are valid with possible problems with the stakeholders. They should have an active role in the development of the system, they must be kept always updated on the temporary results. In the case that they are not satisfied, this should be discovered as soon as possible. Obviously, this is because the more the project proceed the higher is the cost of changes to the project. Every time it is possible, a meeting with stakeholders should be a good idea in order to predict possible issues.

Possbile risks could be introduced by financial issues too. These is a bigger problem, because these kind of issues are generally unpredictable. It's possible that a sudden crisis causes a loss of money for the whole project. This is actually quite common, however, this situation could be solved with a release of less functionalities on the original date of finish, keeping all the other missing parts for later releases or updates of the system.

Another possible problem is the possibility of people quitting the company during the development, as the IT job market is quite flexible. This is why splitting duties and responsibilities across multiple people is mandatory so that there are not single persons in charge of a single task. Problems should rise also from overstimation of knowledge of our team, so it's always better to consider possbile loss of time due to the study of new technologies or improvements to frameworks or programming languages.
One of the biggest possible problem is related to our dependency on external components and APIs. A change in the terms and conditions or on the inputs or outputs of an API itself, could pose serious technical problems. For what concerns the database it is not such a great problem, because there are a lot of vendors and the access methods are more or less standardized, but for what concerns external APIs like the mapping service (Google Maps) or other external routines, it's worth being updated on every kind of changes or updates.
A good habit is to design the code to be as portable as possible and with a great modularity and complete independence between components.

For the persistence of the data, it's a good practice to have redundant copies of the database with the RAID techinque. The source code have to be part of backups too through and should be uploaded everytime on the proper git private repository. Other kind of risks are related to the failure of the hardware installed in the cars. The two car interfaces and their internet connection to the system should be periodically controlled. These kind of maintenence problems and issues should be covered by our Support System that we have shown in our previous documents.

6. Appendix

6.1 Used Tools

- TeXMaker: to create this pdf document
- Gantt Project: to create Gantt diagrams for the schedule and the resource managment

6.2 Working Hours

Last Name	First Name	Total Hours
Larghi	Francesco	10 h
Fantin	Jacopo	6 h