Politecnico di Milano

Software Engineering 2 Project Work

PowerEnJoy Project: Project Plan Document

Version: 1.1

Autohrs:

Jacopo Fantin (mat. 878723)

Francesco Larghi (mat. 876928)

Professors:

Elisabetta Di Nitto

Luca Mottola

January 23, 2017

Contents

1 Introduction **1**

1.1 Revision History . . . 1

1.2 Purpose and Scope . . . 1

1.3 List of Definitions and Abbreviations . . . 1

1.4 List of Reference Documents . . . 2

2 Project Size, Cost and Effort Estimation **3**

2.1 Size Estimation: Function Points . . . 3

2.1.1 Internal Logical Files (ILF) . . . 4

2.1.2 External Interface Files (EIF) . . . 5

2.1.3 External Inputs (EI) . . . 5

2.1.4 External Outputs (EO) . . . 5

2.1.5 External Inquiries (EQ) . . . 5

2.1.6 Overall Estimation . . . 6

2.2 Cost and Effort Estimation: COCOMO II . . . 6

2.2.1 Scale Factors . . . 7

2.2.2 Effort Equation . . . 16

2.2.3 Schedule Estimation . . . 17

3 Project Plan Schedule **19**

4 Resource Allocation **23**

5 Risk Management **27**

6 Appendix **29**

6.1 Used Tools . . . 29

6.2 Working Hours . . . 29

1. Introduction

1.1 Revision History

Version	Date	Author(s)	Summary
1.0	22/01/2017	Francesco Larghi - Jacopo Fantin	First release
1.1	23/01/2017	Francesco Larghi - Jacopo Fantin	Correction of problems

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 $7 \cdot 1 = 7\,FPs$.

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 $3 \cdot 3 + 4 + 3 + 4 = 20\,FPs$.

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 $4 + 5 \cdot 3 + 5 = 24\,FPs$.

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 $3 \cdot 2 = 6\, FPs$ for the EQ function type.

2.1.6 Overall Estimation

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

$$Size = AVC \cdot UFP$$

where AVC is a language-dependent multiplication factor. According to the Function Point Language Table, version 5.0. We assume we employ the factor corresponding to the avarage value for the lower bound estimation instead of the specific lower bound value, as we believe that the resulting range would fit better a realistic situation. Moreover, we obtain a more precise estimated range by doing so. Since we're developing this project on the J2EE platform, this factor equals 46 for the average 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

$$lower\ bound\ Size = 46 \cdot 112 = \mathbf{5152\ LOC}$$

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

2.2 Cost and Effort Estimation: COCOMO II

To compute the approximated effort and the cost in terms of time necessary for the whole application to be implemented, we relied on the algorithm in accordance with the COCOMO II method, a statistical approach to cost estimation for software projects. It consists in derive the effort, expressed in Person-Months, from parameters called Scale Factors and Cost Drivers, and from the LOC number, through a formula that puts in relation all of them to the effort. The effort will be then used among the Scale Factors to derive the estimated time duration, i.e. the schedule estimation, for the project.

The first step when approaching the COCOMO II method is deciding whether we are in a situation of Post-Architecture or Early Design. A tipical case for the first one is when extending an existing software or when developing new software for an already existing software package: the experience with the previously implemented product would come in handy for a better evaluation of the efforts required for the current project. Instead, we would find ourselves in the Early Design case if we didn't have much information about the system architecture. Obviously, the estimation is less accurate in this latter case.

Luckily, we are in the Post-Architecture case, so we'll be able to rely on similar projects previously completed and on the estimations that had been computed for them.

2.2.1 Scale Factors

The second step is determing the right values for the Scale Factors that will make up a part of the final formula used to calculate the effort. While estabilishing the values for Scale Factors the following table provided by the COCOMO II approach has been employed.

Scale Factors	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
PREC SFj	thoroughly unprecedented 6.20	largely unprecedented 4.96	somewhat unprecedented 3.72	generally familiar 2.48	largely familiar 1.42	thoroughly familiar 0.00
FLEX SFj	rigorous 5.07	occasional relaxation 4.05	some relaxation 3.04	general conformity 2.03	some conformity 1.01	general goals 0.00
RESL SFj	little (20%) 7.07	some (40%) 7.07	often (60%) 4.24	generally (75%) 2.83	mostly (90%) 1.41	full (100%) 0.00
TEAM SFj	very difficult interactions 5.48	some difficult interactions 4.38	basically cooperative interactions 3.29	largely cooperative 2.19	highly cooperative 1.10	seamless interactions 0.00
PMAT SFj	SW-CMM Level 1 Lower 7.80	SW-CMM Level 1 Upper 6.24	SW-CMM Level 2 4.86	SW-CMM Level 3 3.12	SW-CMM Level 3 1.56	SW-CMM Level 5 0.00

For each Scale Factor, we tried to decide for our situation what was the best describing case.

Precedentedness: this value is high if similar projects have been implemented previously, so it gives a clue about confidence. Our development team's members haven't worked on such matter yet, therefore we evaluate this item as low.

Development flexibility: this item is about how much flexible the software development process is. As we specified in the Requirement Analysis and Specification Document, we must satisfy several requests that have been formulated by the customer to accomplish the pre-estabilished goals. Although, we might consider the flexibility as nominal as a certain degree of freedom is left by the customer on the system architecture and many of the software functionalities.

Risk resolution: the higher the control we have on the risks and the consciousness of them, the higher the value for this item. For this document we made some assumptions and considerations about risk management. That's why we give a nominal score to the item.

Team cohesion: it evaluates the mood between the teammates and their capacity to cooperate to achieve the common aim. Our group demonstrated a great cohesion and communication abilities to convey information among the members, which began quickly to get along with each other. Thus, we evaluate ourselves with a very high value.

Process maturity: It refers to a method, known as CMM, for evaluating the maturity of a development team. We confer low level on this item, considering that we don't own any experience with the Java EE platform and that we formed a project group just right before this project started.

A table with the final values is shown below:

Scale Driver	Factor	Value
Precedentedness (PREC)	Low	4.96
Development Flexibility (FLEX)	Nominal	3.04
Risk resolution (RESL)	Nominal	4.24
Team cohesion (TEAM)	Very high	1.10
Process maturity (PMAT)	Low	6.00
	Total	**19,34**

Cost Drivers

The next step is evaluating a set of parameters called Cost Drivers, giving an appropriate evaluation to each of them one by one. The evaluations are then translated into the corresponding Effort Multipliers thanks again to tables provided by COCOMO II. These are going to contribute for an other part of the effort formula.

Required Software Reliability (RELY): this parameter measures how much damage a service interruption would cause, mostly from a financial viewpoint. As the potential users would expect a strong reliability on the PowerEnJoy system, we assign a the "high" label to this Cost Driver.

Required Software Reliability (RELY)						
RELY Descriptors	Slight Inconvenience	Low, easily recoverable losses	Moderate, easily recoverable losses	High financial loss	Risk to human life	
Rating	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	0.82	0.92	1.00	1.10	1.26	n/a

Database size (DATA): an estimation of the DB size is given by COCOMO II through the D/P ratio(DB bytes/LOC). Considering that we expect our DB to contain more or less 2GB of data, and remembering the computed number of LOC is in the range of 5150-7500, the resulting D/P ratio falls in the range of 400-582, so DATA assumes a high value.

Database size (DATA)						
DATA Descriptors		D/P<10	10 < D/P <100	100 < D/P <1000	D/P >1000	
Rating	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	0.90	1.00	1.14	1.28	n/a

Product complexity (CPLX): as we tried to keep the software structure and components as simple as possible, we believe the level of complexity should be considered as nominal.

Product complexity (CPLX)						
Rating	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	0.73	0.87	1.00	1.17	1.34	1.74

Required reusability (RUSE): reusability was not a priority for our specifications and was never explicitly requested by the customer, thus RUSE assumes a nominal value.

Required reusability (RUSE)						
RUSE Descriptons		None	Across project	Across program	Across product line	Across multiple product lines
Rating	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	0.95	1.00	1.07	1.15	1.24

Documentation match to life cycle needs (DOCU): the DOCU evaluates whether the project documentation meets the needs of the project life cycle. Since the project documentation is up to date with the project life cycle needs the evaluation will be again set as nominal.

Documentation match to life-cycle needs (DOCU)						
DOCU Descriptons	Many life-cycle needs uncovered	Some life-cycle needs uncovered	Rightsized to life-cycle needs	Excessive for lifecycle needs	Very excessive for life-cycle needs	
Rating	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	0.81	0.91	1.00	1.11	1.23	n/a

Execution time constraint (TIME): Being this Cost Driver the measure of the usage of the CPU processing power, and considering that the service has a strong real-time constraint due to the fact that it is supposed to receive lots of requests each minute, we label TIME with high.

Execution time constraint (TIME)						
TIME Descriptons			<= 50% use of available execution time	70% use of available execution time	85% use of available execution time	95% use of available execution time
Rating	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	n/a	1.00	1.11	1.29	1.63

Storage constraint (STOR): the STOR evaluates the storage constraint that is imposed in a software. We didn't take into acccount particular storage limitation, and we don't expect the application to need that much storage capacity. That's why we will just consider this driver as nominal.

Storage constraint (STOR)						
STOR Descriptons			<= 50% use of available execution time	70% use of available execution time	85% use of available execution time	95% use of available execution time
Rating	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	n/a	1.00	1.05	1.17	1.46

Platform Volatility (PVOL): the higher this parameter, the more frequent will be the updates of the source code. Being PowerEnJoy first of all a mobile application, and considering the fact that we are concerned to release only the essential functions with the first release of the software, we expect PVOL to be in between a nominal and a high value, with major changes on average every 4 months and weekly minor changes.

Platform Volatility (PVOL)						
PVOL Descriptons		Major change every 12 month, minor change every 1 month.	Major change every 6 month, minor change every 2 weeks.	Major change every 2 month, minor change every 1 week.	Major change every 2 weeks, minor change every 2 days.	
Rating	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	0.87	1.00	1.15	1.30	n/a

Analyst Capability (ACAP): ACAP shows the level of ability and efficiency of the team as for the analysis skills. We consider our analysis work as more than sufficient as we came to a deep enough level of comprehension of the problem solved by the app.

Analyst Capability (ACAP)						
ACAP Descriptons	15th percentile	35th percentile	55th percentile	75th percentile	95th percentile	
Rating	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.42	1.19	1.00	0.85	0.71	n/a

Programmer Capability (PCAP): like ACAP regards analysis aspects, PCAP evaluates the programming capacity of the team members. Since the project has not been implemented yet we estimated this Cost Driver looking at the individual teammate's experience and skills on programming, resulting again in a high evaluation.

Programmer Capability (PCAP)						
PCAP Descriptons	15th percentile	35th percentile	55th percentile	75th percentile	90th perc
Rating	Very Low	Low	Nominal	High	Very H
Effort multipliers	1.34	1.15	1.00	0.88	0.76

Personnel Continuity (PCON): The personnel continuity factor indicates how much of the project team will change in one year. It assumes the maximal value in our case, because we are going to end up the project exactly like we started.

Personnel Continuity (PCON)						
PCON Descriptons	48% per year	24% per year	12% per year	6% per year	3% per year	
Rating	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.29	1.12	1.00	0.90	0.81	n/a

Application Experience (APEX): APEX gives an idea of the experience the team has with the application. As we nearly passed 4 months dealing with this software, the suitable value would be something between low and very low.

Application Experience (APEX)						
APEX Descriptons	<= 2 months	6 months	1 year	3 year	6 year	
Rating	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.22	1.10	1.00	0.88	0.81	n/a

Platform Experience (PLEX): this is the the rating of the experience of the group with the development platform, which in our case is Java EE. Therefore we decide to evaluate it as low since none of us is confident with this platform at the time being.

Platform Experience (PLEX)						
PLEX Descriptons	<= 2 months	6 months	1 year	3 year	6 year	
Rating	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.19	1.09	1.00	0.91	0.85	n/a

Language and Tool Experience (LTEX): similar to the PLEX, the LTEX stands for team members experience with the language and tools that have been employed for the project development. We decided to choose a nominal level as an average between our long term experience with the language, Java, and the short term experience with the development tools such as the development platform.

Language and Tool Experience (LTEX)						
LTEX Descriptons	<= 2 months	6 months	1 year	3 year	6 year	
Rating	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.20	1.09	1.00	0.91	0.84	n/a

Use of Software Tools (TOOL): this Cost Driver is useful to understand what kind of developing method and environment has been employed to carry on the project. We had at disposal strong enough software tools that support modern programming and software engineering patterns in an integrated way, so TOOL is evalueted as high.

Use of Software Tools (TOOL)						
TOOL Descriptons	Edit, code, debug	Simple, frontend, backend CASE, little integration	Basic lifecycle tools, moderately integrated.	Strong, mature lifecycle tools, moderately integrated	Strong, mature, proactive lifecycle tools, well integrated with processes, methods, reuse.	
Rating	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.17	1.09	1.00	0.90	0.78	n/a

Multisite development (SITE): SITE takes in consideration the aftermaths of team members living far one from the others, and is a combination of two factors: site collocation and communication support. In our case, the team meets daily or almost daily as we may say the members live in the same metro area. However, they're supported by any kind of modern communication media, first of all chatting services. For this reason we bestow a high label on SITE.

Multisite development (SITE)						
SITE Collocation	International	Multi-city and multi-company	Multi-city or multicompany	Same city or metro area	Same building or complex	Fully collocated
SITE Communications	Some phone, mail.	Individual phone, FAX.	Narrowband email.	Wideband electronic communication.	Wideband electronic communication.	Interactive multimedia.
Rating	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.22	1.09	1.00	0.93	0.86	n/a

Required development schedule (SCED): the parameter measures how much time has been employed by the development team more or less than the nominal required amount of effort in the development of a project. It is evaluated as low due to the fact that a little less time than the nominal one has been spent by the members for PowerEnJoy, mainly because of othersimultaneous pending tasks.

Required development schedule (SCED)						
SCED Descriptons	75% of nominal	85% of nominal	100% of nominal	130% of nominal	160% of nominal	
Rating	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.43	1.14	1.00	1.00	1.00	n/a

Putting together all the results of the analyzed points we obtained the following table:

Cost Drive	**Factor**	**Value**
Required Software Reliability (RELY)	High	1.10
Database size (DATA)	High	1.14
Product complexity (CPLX)	Nominal	1.00
Required reusability (RUSE)	Nominal	1.00
Documentation match to life cycle needs (DOCU)	Nominal	1.00
Execution time constraint (TIME)	High	1.11
Storage constraint (STOR)	Nominal	1.00
Platform Volatility (PVOL)	Nominal-High	1.10
Analyst Capability (ACAP)	High	0.85
Programmer Capability (PCAP)	High	0.88
Personnel Continuity (PCON)	Very High	0.81
Application Experience (APEX)	Low-Very Low	1.15
Platform Experience (PLEX)	Low	1.09
Language and Tool Experience (LTEX)	Nominal	1.00
Use of Software Tools (TOOL)	High	0.90
Multisite development (SITE)	High	0.93
Required development schedule (SCED)	Low	1.14
Total product		**1.1096**

2.2.2 Effort Equation

Finally, we can apply the formulas COCOMO II supplies us with to compute the effort estimation in PM. We first compute the exponent E we will apply later in the effort formula.

$$E = B + 0.01 \cdot \sum_{i=1}^{5} SF_i = 1.1034$$

where B = 0.91 and the 0.01 constant value are empirically measured and in compliance with the COCOMO II algorithm, and SF_i stands for each of the Scale Factors mentioned in the previous paragraphs.

The effort equation according to the method is as follows:

$$Effort = A \cdot Size^E \cdot \prod_{i=1}^{1} 7EM_i$$

where A = 2.94 is a constant in PM/KSLOC, Size is the LOC number found earlier in this chapter expressed in KSLOC, E is the exponent we have computed right above and EM_i stands for each of the Effort Multipliers we dealt with talking about Cost Drivers.

Since for the LOC we provided a lower and an upper bound, so will we do for the effort.

$$lower\ bound\ Effort = A \cdot lower\ bound\ Size^E \cdot \prod_{i=1}^{1} 7EM_i = \mathbf{19{,}903\ PM} \approx 20PM$$

$$upper\ bound\ Effort = A \cdot upper\ bound\ Size^E \cdot \prod_{i=1}^{1} 7EM_i = \mathbf{30{,}134\ PM} \approx 30PM$$

2.2.3 Schedule Estimation

We are now ready to extract the schedule estimation in terms of months needed for the project development. First we need to obtain another exponent F:

$$F = 0.28 + 0.2 \cdot (E - B) = 0.31868$$

Then we can apply the following formula to obtain the estimated duration:

$$Duration = 3.67 \cdot Effort^F$$

Splitting this last one in two, we derive the lower and upper bound for the duration:

$$lower\ bound\ Duration = 3.67 \cdot lower\ bound\ Effort^F = \mathbf{9.52\ months}$$

$$upper\ bound\ Duration = 3.67 \cdot upper\ bound\ Effort^F = \mathbf{10.86\ months}$$

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**

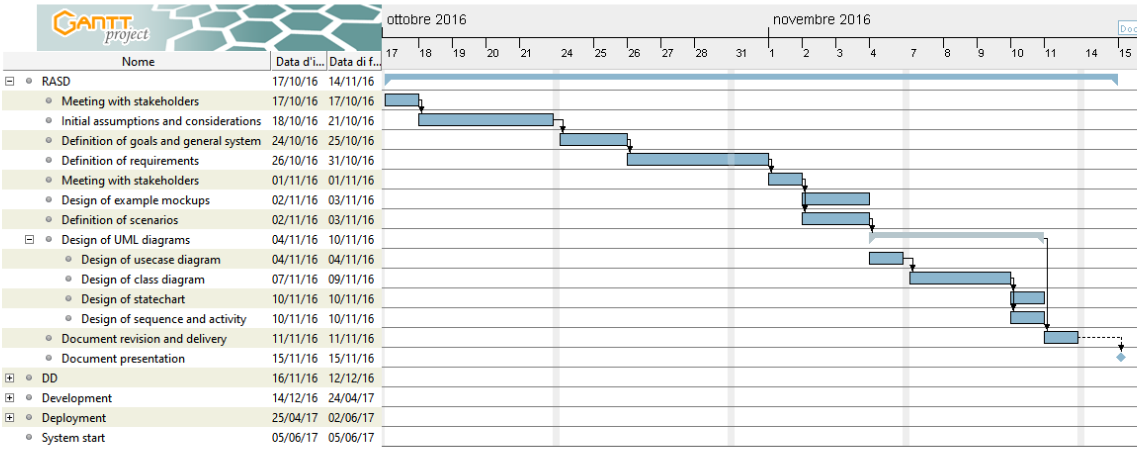

Figure 3.0.1: RASD schedule

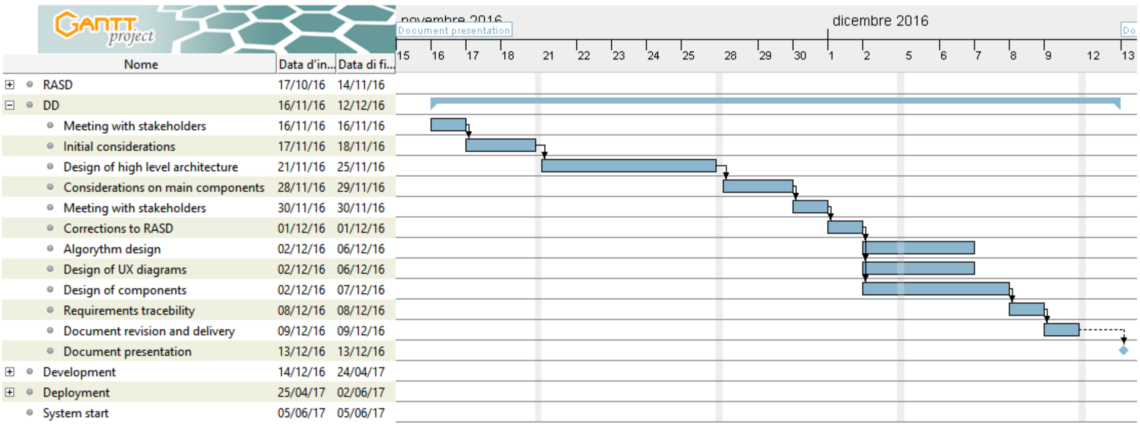

Figure 3.0.2: DD schedule

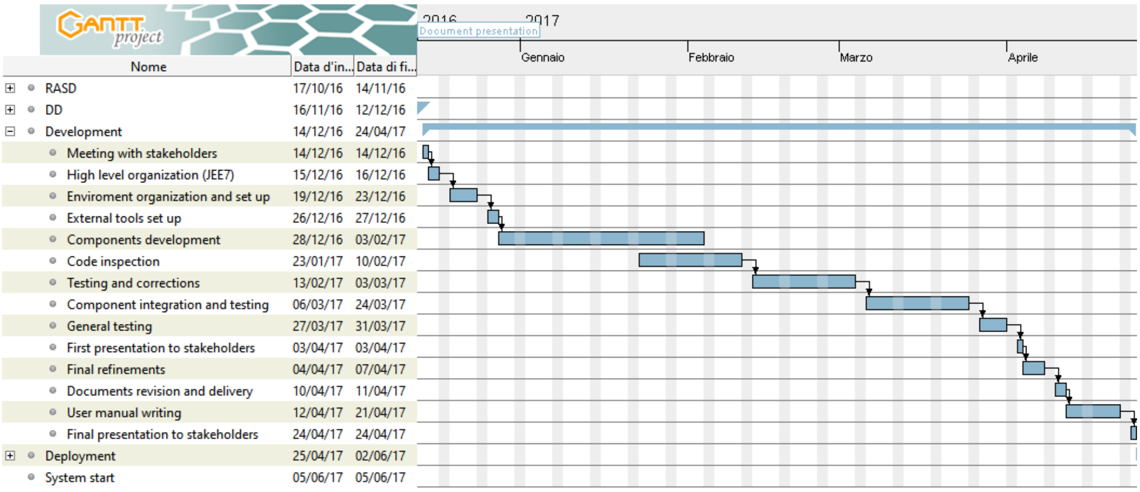

Figure 3.0.3: Development schedule

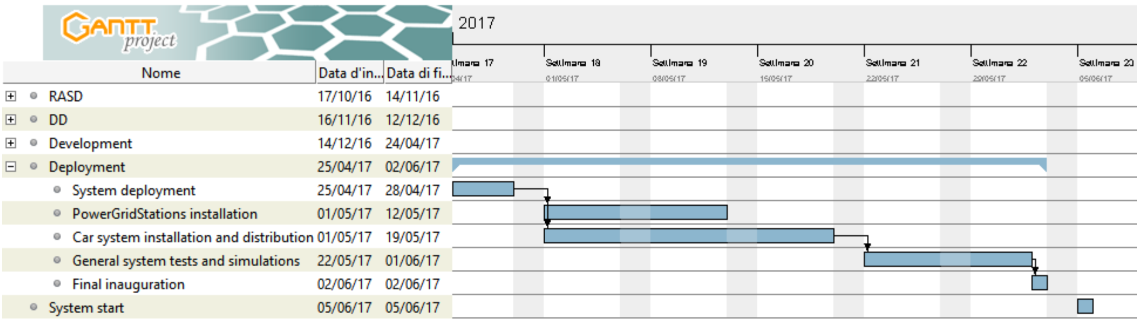

Figure 3.0.4: Deployment and start schedule

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).

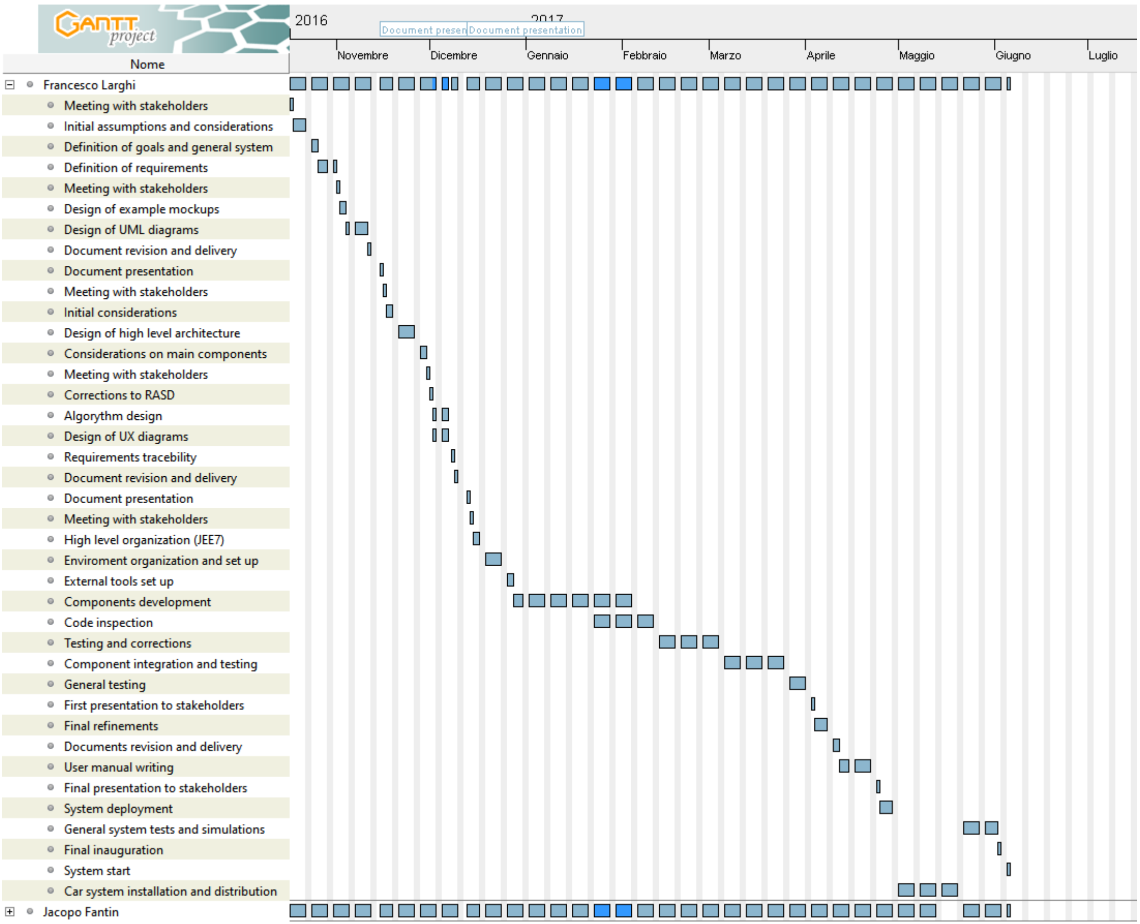

Figure 4.0.1: Rescource allocation for Francesco Larghi

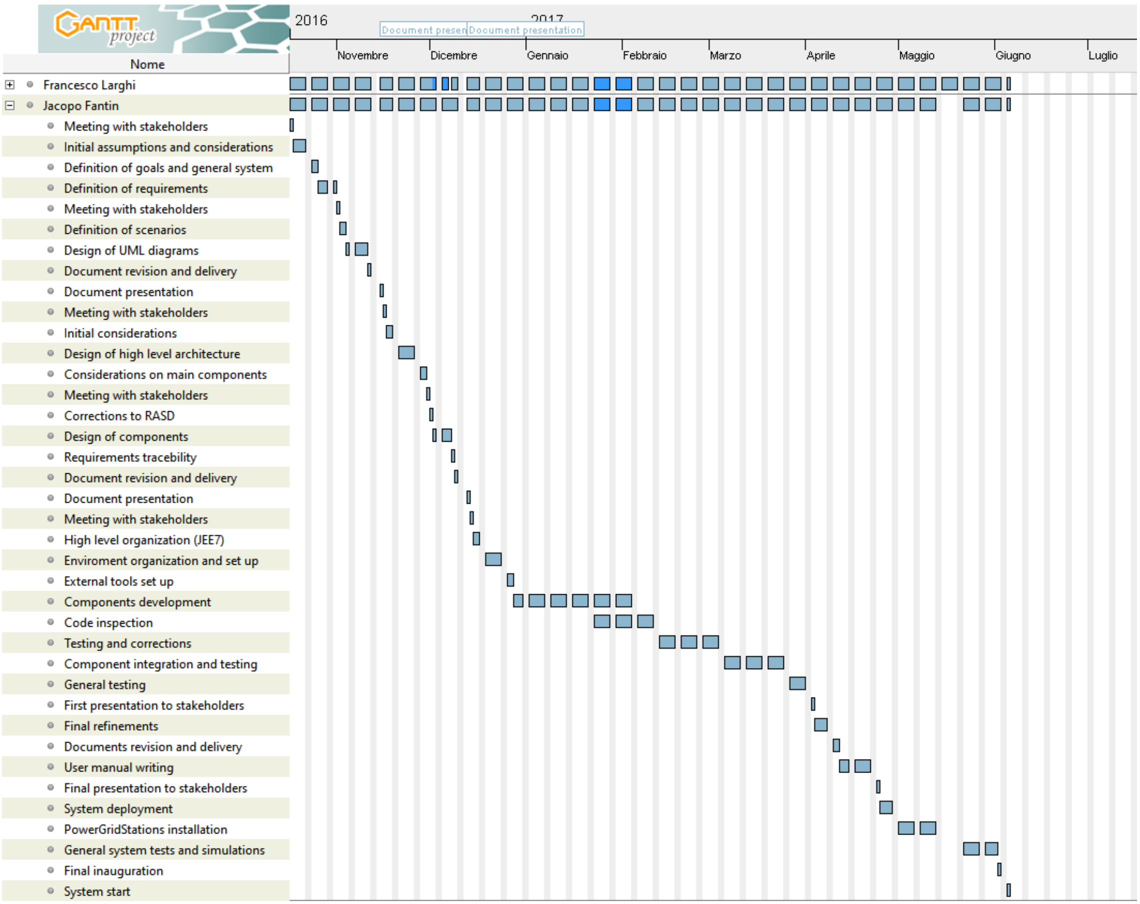

Figure 4.0.2: Rescource allocation for Jacopo Fantin

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 management

6.2 Working Hours

Last Name	First Name	Total Hours
Larghi	Francesco	15 h
Fantin	Jacopo	6 h