Power EnJoy
Project Plan

Niccolo' Raspa, Matteo Marinelli

January 20, 2017

Software Engineering 2 Course Project

Contents

1 Introduction

1.1 Revision History

Version	Date	Author(s)	Summary
1.0	20/01/2016	Niccolo' Raspa, Matteo Marinelli	Initial Release

1.2 Purpose and Scope

The purpose of this document is to provide a reasonable estimate of the complexity of the PowerEnJoy project in terms of the development team effort.

This document is structured as follows:

Chapter 1 Provides general information about the PP document.

Chapter 2 We use Function Point Analysis (FPA) to estimate of the expected size of Power Enjoy in terms of lines of code and of the cost/effort required to actually develop it.

Chapter 3 The Cocomo II estimation process

Chapter 4 Provides a list of all the tasks we have identified and defines a possible schedule to cover all activities.

Chapter 5 Explains how the various tasks are assigned to different members of the group

Chapter 6 Contains an analysis of all kinds of risks that the project could be facing and possible mitigation strategies.

1.3 List of Definitions and Abbreviations

DD: Design Document

RASD: Requirement Analysis and Specification Document

ITPD: Integration Test Plan Document

FP: Function Points.

ILF: Internal logic file

ELF: External logic file.

EI: External Input.

EO: External Output.

EQ: External Inquiries.

Power User: Registered user of the application

External System: Refers to third party systems used in the Power EnJoy application (Payment System, Assistance System, Car-On-Board System)

1.4 List of Reference Documents

Please refer to the following documents, for additional informations on the Power Enjoy System:

- Project rules of the Software Engineering 2 project
- Power Enjoy - Requirement Analysis and Specification Document
- Power Enjoy - Design Document
- Power Enjoy - Integration Testing Plan Document
- Function Point Languages Table - http://www.qsm.com/resources/ function-point-languages-table
- The COCOMO II Model Definition Manual (version 2.1, 1995 – 2000 Center for Software Engineering, USC).

2 Function Points Estimation

The Function Points approach provides an estimation of the size of a project taking as inputs the amount of functionalities to be developed and their complexity. The estimation of the weights is based on the usage of figures obtained through statistical analysis of real projects, taking into account data elements for each record element and each file type.

The following table shows the conversion from weight to FPs:

	Complexity	**Weight**	
Function Type	Low	Avg.	High
ILF	7	10	15
EIF	5	7	10
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6

2.1 Internal Logic Files

Internal Logic Files are homogeneous set of data managed by the application being deveoped.

Data will conventionally be stored in a database whose entities will be User, Vehicles, Safe Area, Plug Station, Location, Reservation and Invoice.

- **User** stores Name, Surname, Driving Licence and other information about people registered to the service. This file type usually has a low complexity since is a basic point of a project, but inside User relation will be stored also the payment method and the login credentials (only costumer type of user is necessary to the system so is useless to separate the login informations). According to this fact, the complexity of User file type will increase to avarage.
- **Vehicle** stores all data about cars managed by the system. Like User, in the Vehicle relation are not present only data to distinguish the cars, but also data about the status og the car (battery, engine, availability). Usually basic relation have low weight but, since it is a complete Vehicle descrition, comlexity will be avarege.
- **Safe Area and Plug Station** are two simple relations since Plug Station is subset of Safe Area. Those relations don't have many attributes, therefore the complexity will be low.
- **Location** is a particular table that store Latitude and Longitude (format not yet known) of users, cars and Safe areas. Even if this table is in relation with at least tre entities, the data to manage are just two, so the comlexity will be low.

- **Reservation** is the core of the system. It need to store all data to identify univocally a reservation and all business data. Since this is the biggest relation, linked with three other tales and full off attribute, the correct weight for this file type is avarage)
- **Invoice** is a table to collect data about the payments at the end of a reservation. The attributes are not so much, nothing relevant is about other attributes, so the complexity will be low.

The following table sum up all ILF in Power Enjoy project with its weight.

File	**Counting Weights**	**FPs**
User	Avg.	10
Vehicle	Avg.	10
Safe Area	Low	7
Plug Station	Low	7
Location	Low	7
Reservation	Avg.	10
Invoice	Low	7
Tot		**58**

2.2 External Interface Files

External Interface Files are homogeneous set of data managed by the application but created elsewhere.

The only set of data respondig to this characteristics is the map showed in the application. It is usefull for a good quality of service and data about may be read from external datasets. Anyway the information exange in order to perform a correct utilization of the map is quite huge and is also necessary a connection and a contract with the owner of those informations, according to this the complexity of Maps file type will be high.

The following table sum up all EIF in Power Enjoy project with its weight.

File	**Counting Weights**	**FPs**
Maps	High	10
Tot.		**10**

2.3 External Input

External Inputs are operation invoked for doing a simple operation on the system with external data.

- **Login** involve just a single entity, the user entity, and the computation is limited just to check the correctness of the Login Credentials. Is reasonable give it a low complexity.

- **Sign up and Profile Management** are similar operations. Both involve only the user entity, computation consists in correctness check and storage. Since the data management is consistend end double copmputed, the complexity of those file types is avarage.

- **Logout** consists in a simple reaction to a logout request, computation is just the closure of the application so is ovious give a low complexity.

- **Search Car** represents the input given by a user who want to reserve a car. This operation start the reserch of available cars according to the position passed as imput. The computation consists in the calculation of the area to search available suggested cars. This operation si not complex but still not so easy to have a low complexity, since require the interaction of some entities using a high amount of data in the calculus. According to this reasoning the complexity will be avarage.

- **Start Reservation** is another main operation of the system. Entities involved are a lot since the relation Reservation need elements about User, Car and Location to create a comlete tuple. Speaking about computation, operatio provide basically the creation of a reservation. The computation indeed is simple but require an high amount of data, according to this reasoning the complexity of Start Reservation is avarage.

- **Car Status Change** involve only the car entity (that is also the one who send the input) and the computation is really simple, indeed consists in a mere update of the car status. Lock and Unlock (for Unlock the input origin is the user) are status managed with different operations because triggered in different situations, but the operation and computation type and size is the same of Car Status Change. As descripted, the complexity of those three operation is obviously low.

- **Cancel and End Reservation** are similar operations, the nly difference is that from an end an invoice generation process starts. The computation is rellay low and consists in the udate of information about reservation and related entities according to the input. Only Reservation entity is directly involved and the computation, as just sed, is simple, so the complexity is low.

- **Ride Details i**nvolve only the reservation entity and the data exange consist in the receiving of the details of a ride (passengers, time, battery,...) from the car. The computation consists in the storage and the invocation of the calcuation of the invoice. All this operation are quite simple, so the complexiti of this file type is low.

- **Malfunctioning Fixed** is an input received from the Assistance service. It involves directly only Assistance entity and the computation is just a status change in the Car relation. All operation needed are basically not so much and all simple, so the complexity is low.

- **Payment Succesful** or not are opposite input from the Payment Service. The entity involved is the Payment entity and the computation is to set or rest the Paiment pending status of the user. This operation are really simple and do not require expensive computaion, so the complexity is low.

The following table sum up all External Input in Power Enjoy project with its weight.

Operation	**Counting Weights**	**FPs**
Login	Low	3
Profile Management/Signup	Avg.	2x4
Logout	Low	3
Select Car	Avg.	4
Start Reservation	Avg.	4
Cancel/End Reservation	Low	2x3
Car Status Change/Unlock/Lock	Low	3x3
Car Malfunctioning	Low	3
Ride Details	Low	3
Malfunctioning Fixed	Low	3
Payment Succesfull or not	Low	2x3
Tot.		**52**

2.4 External Inquiries

External Inquiries are operation that involves both input and outpt, mainly for retrieving information from the system.

The only External Inquiry actually imagined for the system is the retrieving of Reservations History and Payment History. An User must have the possibility to controll his reservation and/or payment history, this operation involve only the reservation entity or the invoice entity and provide all data usefull for the user. Since the entity involved is one and the data requested are not so much, the complexity of those operations is low.

The following table sum up all External Inquiries in Power Enjoy project with its weight.

Operation	**Counting Weights**	**FPs**
Reservations History	Low	3
Payment History	Low	3
Tot.		**6**

2.5 External Output

External Output are system operation producing data for the external environment.

- **Not Valid Informations Notification** about the wrong format or content of the information inserted during a Signup or a Profile management involves only User entity. Since this output is computed as a reaction to the check, the complexity is low because the computation is just the reading of the check result and the creation of the notification.

- **Wrong User or Password Notification** is affected by the same reasoning. Is a notification roduce after a check, so the computation is simple and the complexity is low.

- **Available Car Set** is the list of available cars requested by the user, the computation consists in the research of the car in the area calculated from the Car Selection input and involve reservation, user, cars and location entities at least. After the research the list must be created and sended as output. This set of operation could be quite complex and, togheter the fact a lot of entities are involved, the complexity of this operation is high.

- **Reservation Confirmed** notifies about the succesfully creation of a reservation in the database and in the logic. It involve only reservation entity and no complex computation are necessary since the output is created as a tirgger, so the comlexity of this operation is low.

- **Unlock Notification** is a particular operation. When an unlock request is performed, this operation is invoked. It involve User, Reservation, Car and Location. The computation consists in the check of the distance between the user and the car and according to this is produced an output that notify the car to unlock itself or the user that unlock is impossible since the distance is too much. Since the high amount of entityes involved and the particular computation, the complexity of this operation is high.

- **Invoice Notification** is invoked when a car is turned off in a safe area. It involve the Invoice, the User and the Reservation entities and compute the amount to pay according the data received about the ride from the car. This operation, since involve four entities and a adequate quantity of data, has avarage complexity.

- **Pending Payment Notification** is a operation that notify an user about the impossibilitatio to reserve a car because a pending payment. The entity involved is only user, since the data are stored in user relation, and the computation consist just in a ceck and the notification, so the complexity is low.

- **Require assistance** send a request to the assistance service. Only assistance entity is involved and the computation si really simple because is just the reception of an assistance request from a car and the production of the output. The complexity is low.

- **Payment Request** has the same characteristics of Require Assistance, but the target of the output is the Payment service and the entity involved is the payment. The data involved in the exchange are some more but not enough to change the complexity that is also low.

The following table sum up all External Outputs in Power Enjoy project with its weight.

Operation	**Counting Weights**	**FPs**
Not Valid Credential Notification	Low	4
Wrong User or Password Notification	Low	4
Available Cars Set	High	7
Reservation Confirmed	Low	4
Unlock Notification	High	7
Invoice Notification	Avg.	5
Pending Payment Notification	Low	4
Require Assistance	Low	4
Payment Request	Low	4
Tot.		**43**

2.6 Results

According to QSM - Function Point Languages Table, the following holds for J2EE. As an average:

$$\frac{SLOC}{FPs} = 46$$

ands upper bound:

$$\frac{SLOC}{FPs} = 67$$

The *sum of all FPs is 168,* consequentially

$$SLOC_{Average} = 46x168 = 7728$$

$$SLOC_{upperbound} = 67x168 = 11256$$

3 Cost and effort estimation: COCOMO II

3.1 Scale Drivers

Some of the most important factors contributing to the duration and cost of a project are the Scale Drivers. Each of said drivers describes the project itself and determines the exponent used in the Effort Equation. More precisely, the Scale Drivers reflect the non-linearity of the effort with relation to the number of SLOC. Each scale driver has a range of rating levels from Very Low to Extra High.

In order to evaluate the values of the scale drivers, we refer to the following official COCOMO II table:

Scale Factor values, SFj, for COCOMO II Models:

Scale Factors	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
PREC	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	thoroughly familiar
SFj	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	generals goals
SFj	5.07	4.05	3.04	2.03	1.01	0.00
RESL	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
SFj	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
SFj	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	Level 1 Lower	Level 1 Upper	Level 2	Level 3	Level 4	Level 5
SFj	7.80	6.24	4.68	3.12	1.56	0.00

A brief description for each scale driver:

- **Precedentedness**: it reflects the previous experience of our team with the development of large scale projects. Since we are not expert in the field, but only in programming and also not togheter, this value will be low.
- **Development flexibility**: it reflects the degree of flexibility in the development process with respect to the external specification and requirements. Since there are very clear requirements on the functionalities but is not mandatory following them strictly and nothing specific is stated as for the technology to be used, this value will be high.
- **Risk resolution**: reflects the level of awareness and reactiveness with respect to risks. The risk analysis we performed is quite extensive, so the value will be set to very high.
- **Team cohesion**: it's an indicator of how well the team members know each other and work together in a cooperative way. Our team met for the first time at the beginning of the work, all memebers has different background and come from different universities but everyone has experience in team working, so the value will be nominal.

- **Process maturity**: although we surerly have some problems during the development of the project, the goals will be successfully achieved. Since this is our first project of this kind, this value is set to level 3.

The results of our evaluation is the following:

Scale Driver	Factor	Value
Precedentedness (PREC)	Low	4.96
Development flexibility (FLEX)	High	2.03
Risk resolution (RESL)	Very High	1.41
Team cohesion (TEAM)	Nominal	3.29
Process maturity (PMAT)	Level 3	3.12
Tot.	$E = 0.91 + 0.01 \sum_i SF_i$	1.0581

3.2 Cost Drivers

Cost Drivers appear as parameters of the effort equation reflecting characteristics of the developing process and acting as multiplication factors on the effort needed to carry out said project.

The early design version of the COCOMO II model will be used for the effort analysis since it is carried out at the beginning of the life-cycle of the project, before the writing of the RASD and the DD.

From this point of view, clear information about the architecture are still not available and the team is exploring and evaluating different architectural alternatives.

In order to conduct the analysis of said cost drivers, guidelines given by the COCOMO Model Definition Manual have been followed. Each cost driver of the post-architecture approach has a rating that is considered as a numerical value during the early design analysis (Very Low = 1, Extra High = 6). Ratings related to the Post-Architecture cost drivers corresponding to an Early Design one will be summed and converted into rating levels using the provided tables.

Personnel Capability is the union of Analyst Capability (ACAP) factor, the Programmer Capability (PCAP) factor and the Personnel Continuity (PCON) factor and represent the potential expressed by the work team.

- **Analyst Capability (5)**: Members of the team has an amazing analysis skill. Discussion about the projec has alredy covered an incredible number of situations, also watching limit situation like parking in dangerous location and similar. According to this, the Analyst Capability is set to Very High.

- **Programmer Capability (3)**: We have not implemented the project, so this parameter is just an estimation; however not the totally of the group is really skilled in programming, even if some excellence are present in the team. This small lack bring this cost parameter to nominal.

- **Personnel continuity (2)**: All member of the team can not consider the project as the main occupation. Everyone has its own business in univerity and also outside. A relevant fact is the mobility, no one of the team is originary from Milan, the developement place, so movements cause obviously discontinuities. This parameter is set to low.

Sum of ACAP, PCAP, PCON Ratings	5,6	7,8	9	**10,11**	12,13	14+
Rating Levels	Very Low	Low	Nominal	**High**	Very High	Extra High
Effort Multipliers	1.62	1.26	1.00	**0.83**	0.63	0.50

Product Reliability and Complexity is the union of Required Software Reliability (RELY), Database Size (DATA), Product Complexity (CPLX), and Documentation Match to Life-cycle Needs (DOCU).

- **Required Software Reliability (4)**: PowerEnjoy service is provided by a private company as an offer to the population of Milan, anyway a malfunctioning of the service may provoke an huge costumers loss, and consequentially an high finantial loss, for the problem related to the mobility but also for the problem that may occur to open reservation that will be no longer supperted by the server. According to this reasoning the Reliability should be high.

- **Database size (4)**: This measure considers the effective size of our database. We don't alredy know the real dimension, but our estimation given the tables and fields we have is to reach a 4GB database (seven tables with a lot of attributes but mainly that have to contain an huge amount). Since it is distributed over 7.500-11.000 SLOC, the ratio D/P (measured as testing DB bytes/program SLOC) is between 292 and 460, resulting in the DATA cost driver being high.

- **Product complexity (4):** The product format is quite complex but not new to the world market. We will take example from alredy existing projects in circulation, obviousli without surpassing the limit of illegality. According to this fact, a normal very high cost will be reduced to an high cost.

- **Documentation match to life-cycle needs (2)**: This parameter describes the relationship between the documentation and the application requirements. Since our project is linked with external services, two of them in a designing phase like this one (Car On Board System and Assistance Service), could be possible that some life-cycle needs escape from our foreseen, so the DOCU cost driver is set to low as a precaution.

Sum of RELY, DATA, CPLX, DOCU Ratings	7,8	9-11	12	**12-15**	16-18	19+
Rating Levels	Very Low	Low	Nominal	**High**	Very High	Extra High
Effort Multipliers	0.60	0.83	1.00	**1.33**	1.91	2.72

Required reusability In our case, the reusability requirements are limited in scope to the project itself or similar, so the RUSE cost driver is set to nominal.

RUSE Descriptors	None	**Across Project**	Across Program	Across Product Line	Across Multiple Product Lines
Rating Level	Low	**Nominal**	High	Very High	Extra High
Effort Multipliers	0.95	**1.00**	1.07	1.15	1.24

Platform Difficulty represent the intrinsic complexity of the platform expressed by Execution time constraint (TIME), Storage constraint (STOR) and Platform Volatility (PVOL).

- **Execution time constraint (5):** This parameter describes the expected amount of CPU usage with respect to the computational capabilities of the hardware. Power Enjoy service software must be able to manage an huge amount of data quickly and efficiently, without suffering delays and performing complex operations, all simoultaneously. Our expectance is that its CPU usage will be very high.
- **Storage constraint (3):** This parameter describes the expected amount of storage usage with respect to the availability of the hardware. As current disk drives can easily contain several terabytes of storage, this value is set to nominal.
- **Platform Volatility (3):** For what concerns the core system, we don't expect our fundamental platforms to change very often. Also the client applications may require at least a major release once every six months to be aligned with the development cycle of the main mobile operating systems. May be possible also the insertion of different type of reservable

vehicles and this may need a sequence of updates. For this reason, this para6+meter is set to nominal.

Sum of TIME, STOR, PVOL Ratings	8	9	**10-12**	13-15	16+
Rating Levels	Low	Nominal	**High**	Very High	Extra High
Effort Multipliers	0.87	1.00	**1.29**	1.81	2.61

Personnel Experience is evaluated with the parameters Application Experience (APEX), Platform Experience (PLEX) and Language and Tool Experience (LTEX).

- **Application Experience (2):** We have some experience in the development of Java applications, basically just one, but we never tackled a Java EE system of this kind. For this reason we're going to set this parameter to low.
- **Platform Experience (3):** We don't have any experience with the Java EE platform, but we have some previous experience with databases, user interfaces and server side development. For this reason, we're going to set this parameter to nominal.
- **Language and Tool Experience (2):** All team members are better skilled in C (all version), but Java seems to be a better choice to develope the project. Also experience with the tools is basic, so this parameter will be set to low.

Sum of APEX, PLEX, LTEX Ratings	5,6	**7,8**	*9*	10,11	12,13	14+
Rating Levels	Very Low	**Low**	Nominal	High	Very High	Extra High
Effort Multipliers	1.33	**1.22**	1.00	0.87	0.74	0.62

Facilities are a combination of Usage of Software Tools (TOOL) and Multisite development (SITE).

- **Usage of Software Tools (5):** Our application environment is complete and perfectly integrated, so we'll set this parameter as Very high.

- **Multisite development (4):** All the team live in Milan but everyone have to go back in his hometown sometimes, anyway we have collaborated relying hugely on wideband Internet services including social networks and emails. For this reason, we're going to set this parameter to high.

Sum of TOOL, SITE Ratings	3	4,5	6	7,8	**9,10**	11+
Rating Levels	Very Low	Low	Nominal	High	**Very High**	Extra High
Effort Multipliers	1.30	1.10	1.00	0.87	**0.73**	0.62

Required Development Schedule: Although our efforts were well distributed over the available development time, the definition of all the required documentation took a consistent amount of time, especially for the requirement analysis and the design phases. For this reason, this parameter is set to high.

SCED Descriptors	75% of nominal	85% of nominal	100% of nominal	**130% of nominal**	160% of nominal
Rating Levels	Very Low	Low	Nominal	**High**	Very High
Effort Multipliers	1.43	1.14	1.00	**1.00**	1.00

Final Results

Cost Driver	**Factor**	**Value**
Personnel Capability (PERS)	High	0.83
Reliability and Complexity (RCPX)	High	1.33
Required reusability (RUSE)	Nominal	1.00
Platform Difficulty (PDIF)	High	1.29
Personnel Experience (PREX)	Low	1.22
Facilities (FCIL)	Very High	0.73
Required Development Schedule (SCED)	High	1.00
Tot.	EAF $= \prod_i C_i$	**1.23**

3.3 Effort Equation

The formula $Effort = A \cdot EAF \cdot KSLOC^E$ estimates the effort in *Persons-Months (PM)*.

The Parameters are:

- $EAF = Q_i C_i = 1.23$ derived from the *cost drivers analysis*.
- $E = 0.91 + 0.01 \times Q_i SF_i = 1.0581$ derived from the *scale drives analysis*.
- $KSLOC = 7.728$ and $KSLOC = 11.256$ (Kilo Source Lines of Code estimated via *FPs analysis*).
- $A = 2.94$

Therefore the result is:

$$Effort_{average} = 31.47PM$$

$$Effort_{upperbound} = 39.68PM$$

3.4 Schedule Estimation

Now we calculate the duration of the project, with the parameter $B = 0.91$ according to COCOMO II specifications with the formula:

$$Duration = 3.67 \cdot (Effort)^{0.28+0.2(E-B)}$$

- Using the *average value* of effort

  $$Duration = 10,68 months$$

  corresponding to:

  $$\frac{Effort}{Duration} = 2,95 developers$$

- Using the *upperbound value* instead, we get

  $$Duration = 12,47 months$$

  corresponding to:

  $$\frac{Effort}{Duration} = 3,46 developers$$

Our team is composed by just 2 people, one less than the suggested number. Therefore the *real estimate developement time* will be between $\frac{31.47PM}{2people} = 15.74 months$ and $\frac{39.68PM}{2people} = 19.84 months$.
Those results are reasonable according to the results of the Duration equation, since the suggested developers are 3 and we are just in 2.

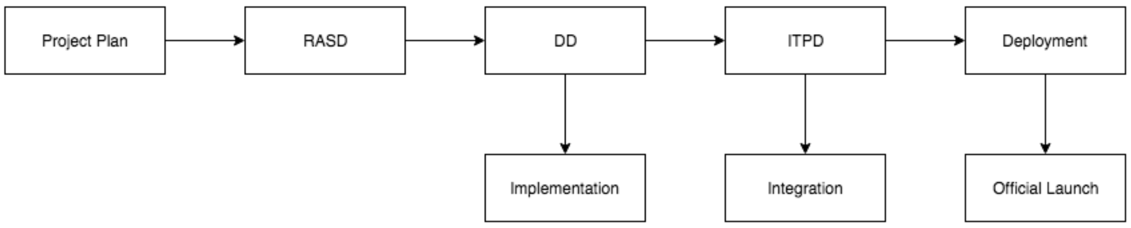
4 Task Scheduling

In this chapter we're going to provide a list of all the tasks we have identified while in the section we're going to explain how they are assigned to each team member. We pretend that we have written this document at the beginning of the project.

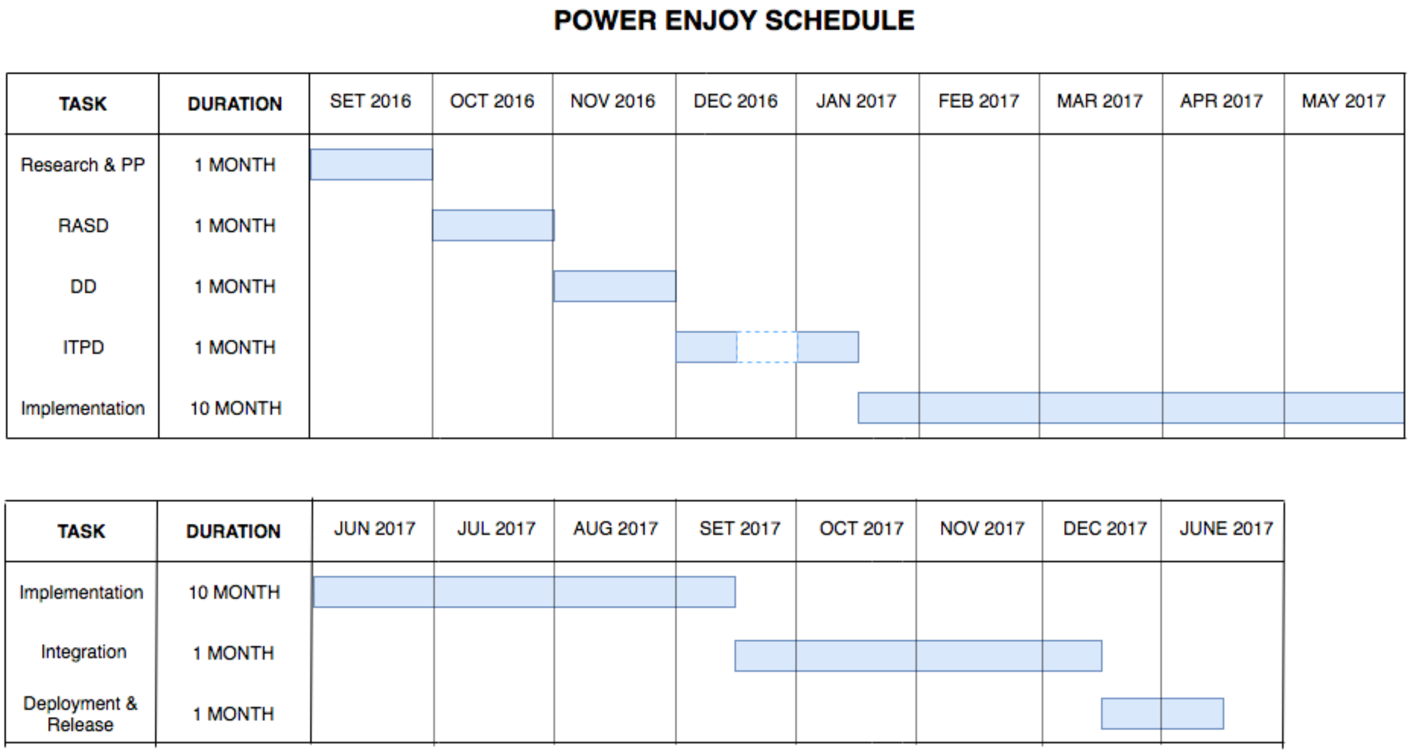
The main tasks of which the PowerEnJoy project are the following:

1. **Research and Project Plan (PP) -** Feasibility Study and produce the project plan document to estimate the complexity of the project;
2. **Requirements Analysis and Specification Document (RASD)** - deliver a document containing the description of all goals, domain assumptions, functional and non-functional requirements for the project;
3. **Design Document (DD)** - deliver a document describing the architectural design of the software system to be produced;
4. **Integration Test Plan Document** (ITPD- deliver a document containing the strategy to perform integration testing on the components of the system;
5. **Implementation** - implement the indipendet part of the software product, perform unit tests and code inspection;
6. **Integration and Testing** - integration of all the software components of the project and testing of the different subsystem as well as the final result;
7. **Deployment & Official Launch** - prepare the software for deployment, prepare market strategy and implement some feedback mechanism for the users;

The next diagram highlights the sequence and the dependencies among this different tasks.



The following is a Gantt chart to describe the chosen schedule for the project:



POWER ENJOY SCHEDULE

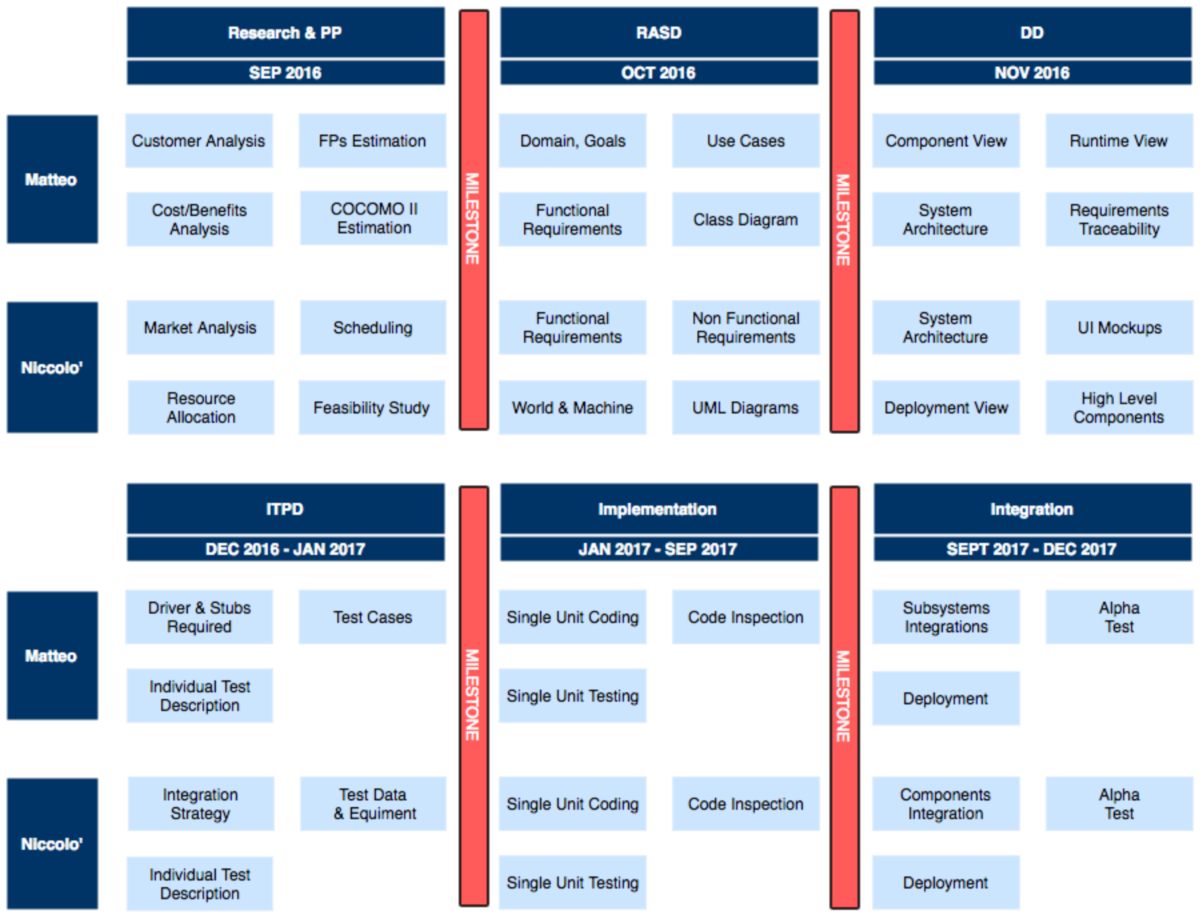
TASK	DURATION	SET 2016	OCT 2016	NOV 2016	DEC 2016	JAN 2017	FEB 2017	MAR 2017	APR 2017	MAY 2017
Research & PP	1 MONTH	▨								
RASD	1 MONTH		▨							
DD	1 MONTH			▨						
ITPD	1 MONTH				▨	▨				
Implementation	10 MONTH					▨	▨	▨	▨	▨

TASK	DURATION	JUN 2017	JUL 2017	AUG 2017	SET 2017	OCT 2017	NOV 2017	DEC 2017	JUNE 2017
Implementation	10 MONTH	▨	▨	▨	▨				
Integration	1 MONTH				▨	▨	▨	▨	
Deployment & Release	1 MONTH							▨	▨

5 Resource Allocation

In this section we're going to allocate the different subtask to each team member. Since the team is only composed of two members the cooperation in this project we'll be easier to manage rather than having teams of people. Even if most activities will involve contribution from both team members, it's important to allocate macro-activities. In this way each team member knows which are his responsabilities and understand his role on the development of the project. This separation will also semplify cooperation in tasks and avoid possibile misunderstandings.
The following table describe in detail the subdivision of every task in different activities and who will be responsible for it. We avoid going too much in details to leave freedom to each member to choose the best strategy to complete each assignment.

Suggested Allocation



	Research & PP			RASD			DD	
	SEP 2016		MILESTONE	OCT 2016		MILESTONE	NOV 2016	
Matteo	Customer Analysis	FPs Estimation		Domain, Goals	Use Cases		Component View	Runtime View
	Cost/Benefits Analysis	COCOMO II Estimation		Functional Requirements	Class Diagram		System Architecture	Requirements Traceability
Niccolo'	Market Analysis	Scheduling		Functional Requirements	Non Functional Requirements		System Architecture	UI Mockups
	Resource Allocation	Feasibility Study		World & Machine	UML Diagrams		Deployment View	High Level Components

	ITPD			Implementation			Integration	
	DEC 2016 - JAN 2017		MILESTONE	JAN 2017 - SEP 2017		MILESTONE	SEPT 2017 - DEC 2017	
Matteo	Driver & Stubs Required	Test Cases		Single Unit Coding	Code Inspection		Subsystems Integrations	Alpha Test
	Individual Test Description			Single Unit Testing			Deployment	
Niccolo'	Integration Strategy	Test Data & Equiment		Single Unit Coding	Code Inspection		Components Integration	Alpha Test
	Individual Test Description			Single Unit Testing			Deployment	

6 Risks Management

In this section we're going to list all the realistic risk and possibile threats that we may encounter during the development of the Power Enjoy System. Since those risks might impact the development of the Power Enhoy System in different ways we have split them in three different sections: project, technical and business risks.

In order to proactive and mitigate the damage, for every risk a mitigation strategy will be proposed to try to minimize the consequences.

6.1 Project Risks

Project risks pose a threat to the project plan and make the project schedule slip, increasing the overall costs. The following risks have been identified:

Risk	Mitigation Strategy
Changing requirements	During the development of the project the requirements can change unexpectedly. This kind of risk cannot be prevented, but it can be reduced by a clear traceability information to assess requirements change impact and maximizing information hiding in the design.
Deadlines not met	It can occur that the project requires more time than expected to be carried out. In this case only some functionalities will be offered in a first release while less essential features will be developed later.
Personnel shortfall	Since there are only 2 team member if for any reason (key staff are ill, team member quits, etc.) any of the team member is forced to leave the project team, the development process will inevitably suffer from huge delays and the risk will result in an impossibility to meet the defined deadlines. This can be countered, in case of social issues, in a preemptive way by valuing individual work and creating a positive work environment.

6.2 Technical Risks

They threaten the quality and timeliness of the software to be produced. The following risks have been identified:

Risk	Mitigation Strategy
Data Loss	A loss of the whole source code, or significant parts would be a disaster. This issues is quite easy to tackle, though, by implementing appropriate versioning systems and backup techniques distributed over multiple, redundant locations.
Defective components of the car system	It can occur that the project requires more time than expected to be carried out. In this case only some functionalities will be offered in a first release while less essential features will be developed later.
Payment System Fails	If for any reason the agreement with the external payment system will cease we must find another external system to substitute it. To facilitate this process our payment handler must be able to abstract from a specific vendor and be able to work with any different payment managers.
Software to difficult to use for average customer	In this we may rethink the UI and the UX trying to leave the business logic as it is.
Scalability Issues	If for any reason the power enjoy couldn't handle the traffic (e.g. database used in the system cannot process as many transactions per second as expected) we may acquire more and more performant machine and spend extra time analyzing and optimizing the bottlenecks for the system.

6.3 Business Risks

Economical risks jeopardize the whole software product threatening its viability.
The following risks have been identified:

Risk	Mitigation Strategy
Local regulation and policies	Some risks could derive from the change of the regulations regarding car sharing, in this case we could find small local goverments that agree with our eco-friendly mission.
Competitors	To have competetive advantage againts competitors that offers similar services, we might leverage our eco-friendly mission and provide more appealing functionality and prices.
Acceptance of the system from its intended users	In case of partial or non acceptance of the system from its intended users we could organize a marketing event to advertise the service while organizing focus groups on potential costumers.
Taxi Driver Protests	We may expect some protest from Taxi Services like for the Uber Company. To mitigate this scenario we might ask support to local governments to help settle this situation.

7 Effort Spent

- Niccolo' Raspa: 12 Hours
- Matteo Marinelli: 18 Hours