Project Planning

Software Engineering 2

January 22,2017

POLITECNICO
MILANO 1863

Authors:

Claudio Salvatore	Arcidiacono	Matr 879109
Antonio	Di Bello	Matr 878786
Denis	Dushi	Matr 879115

Contents

1 Introduction

1.1 Revision History

1.2 Purpose and scope

This is the **Project Plan Document** for the **PowerEnJoy** system. The purpose of this document is to estimate the size, the complexity and the cost of the project, to identify what are the critical and most difficult tasks, to identify some possible risks proposing some solutions and to arrange an approximate schedule for the project team work. In the following section we will first use the function points approach in order to estimate the size of project; identifying the biggest and most crucial function that the system has to provide, highlighting what are the Internal and External logic Files (ILF and ELF), the external inputs,outputs and inquiries (EI, EO and EQ) and what are their function points. Then we will estimate the cost of the project in terms of lines of code and person/month using the Cocomo II approach. The document continue with the schedule of the spotted tasks and with the allocation of our resources on the different tasks. In the last part we will focus our attention on some of the possible risks to be prevented and we will explain some possible solutions.

1.3 Definitions,Acronyms,Abbreviations

1.3.1 Definition

Function Points:A function point is a "unit of measurement" to express the amount of business functionality an information system (as a product) provides to a user. Function points are used to compute a functional size measurement (FSM) of software.
Cocomo II: Constructive Cost Model (COCOMO) is a procedural software cost estimation model developed by Barry W. Boehm. The model parameters are derived from fitting a regression formula using data from historical projects.
Guest: is an user that is not registered yet to PowerEnJoy.
Client: is an user that has completed successfully the registration procedure.
Logged client: is a client that has logged into PowerEnJoy.
Blocked client: is a client that is not allowed to log in to system because he has some debits with PowerEnJoy.
User: a client or a guest.
Virtuous behavior: the client has a "Virtuous behavior" if he performs one or more of the following actions:

- transport at least other two passengers into the car.
- leaves the car with less than 50% of the battery empty.
- plugs the car into the power grid of a charging station.

Rent: defines the period of time that starts when the logged client opens the car and ends when the door of the car are locked due to a "end rent" request of the logged client.
Pin: a personal sequence of four numbers given by the system after the registration, necessary to start the engine.
Map Information: denotes the following information:

- location of available cars.
- location of charging areas.
- for each charging area the number of available charging spots.
- boundaries of safe areas.

Not valid data: syntactically incorrect data (e.g. mail not in the format local-part@domain).
Credentials: email and password used to log into PowerEnJoy.
Valid Credentials: email and password related to a registered user.

Car information Menu: displayed when a client selects a car on the map. In this menu are displayed the information about the car (remaining charge percentage and license plate number) and a button that can be used to reserve the car.
Safe Areas: areas in which is permitted to the client to end the rent and leave the car.
Valid Payment Information: Credit or prepaid card with at least a minimum amount of money.

1.3.2 Acronyms

- **RASD:** requirements analysis and specifications document
- **DD:** design document
- **ILF:** Internal Logic Files
- **EIF:** External Interface Files
- **EI:**External Inputs
- **EO:** External Outputs
- **EQ:**External Inquiries
- **JEE:**Java enterprise edition

1.3.3 Abbreviations

1.4 Reference Documents

- RASD Document
- Design Document
- Assignment AA 2016-2017.pdf
- Project planning example document.pdf

2 Project size, cost and effort estimation

In the section below we will use function points and COCOMO II approaches in order to provide an estimation of the expected size, cost and required effort for **PoweEnJoy**.

2.1 Size estimation: function points

The size estimation will be estimated using the **Function Points** approach, spotting at first what are the main functionalities of our system and how many lines of code we aspect to be written in JEE. According to the function points approach, the LOC can be estimated using the following formula:

$$LOC = FP \cdot AVC$$

where:

- LOC = Lines of code
- FP = Function Points
- AVC = language specific constant, for JEE is between 46 for an lower bound and 67 for an upper bound

In order to estimate the overall function points, this approach needs a first classification of our main files into 2 categories:

- Internal Logic files : homogeneous set of data used and managed by the application .
- External Interface files:homogeneous set of data used by the application but generated and maintained by other applications.

and a classification of the main functionalities into 3 main categories:

- External Input: Elementary operation to elaborate data coming from the external environment.
- External Output:Elementary operation that generates data for the external environment.
- External Inquiry:Elementary operation that involves input and output, Without significant elaboration of data from logic files.

Once we have identified the most relevant functions of our system, we attribute a complexity to each of them and based on the category to which it appertain we assign some function points. The function points have been obtained from statistical studies of past projects. The weights that we will use for each category will be:

	Complexity Weight		
Function Type	*Low*	*Average*	*High*
Internal Logic Files	7	10	15
External Interface Files	5	7	10
External Input	3	4	6
External Output	4	5	7
External Inquiries	3	4	6

2.1.1 Internal Logic Files (ILF)

PowerEnJoy need to stores a big amount of data about users, cars and his assets, so it will use several ILFs. Here below we will see in detail the most relevant.

- First of all the system will store data about its users (name, surname, email, address, fiscal-code, driving license code and picture , ID number, cellphone number) and since this entity will have several attributes and it will contain around 200.000 entries (considering that the system will run in a city like Milan, where other competitors have similar numbers) we can attribute to this ILF an **Average** complexity.

- For each client the system will store data about his payment information, but since this entity has few attributes and it is related 1 to 1 just with the client data, and since these data will be inserted and uploaded rarely we can consider its complexity as **Low**.

- For each client we will need to store his login data, for the same reasoning as above we will consider this entity complexity as **Low**.

- The main function of our system is to allow users to reserve and rent cars so the main data that we will store will concern reservations and rents. Assuming that an average client will do 4-5 rents we estimate that the system must be able to store 800.000 - 1M entries. During the day there are some hours were our users will use the system more than during others so we will have a huge traffic of insert and delete that our system must be able to handle in parallel. Furthermore this entities are linked with a lot of other structures in our system so, given the nature of them we can estimate their complexity as **High**.

- The system also need to store information about cars, an entity that will be updated pretty often and that has several relations with other structures; but we assume that the system will handle between few hundreds and a thousand cars (so entries); compared with other entities they aren't that much, so we can estimate the complexity of the ILF as **Low**.

- The system has to provide a map that will contain information about cars, were to park them and where to charge them so it has to store information about safe areas and charging stations. Since these two entities contain few entries (around one hundred charging stations and tens of safe areas) and that these tables will not be updated very often we can consider their complexity as **Low**.

The overall estimation for ILF is summarized in the following table:

ILF	Complexity	FPs
Clients data	Average	10
Payment Information data	Low	7
Login data	Low	7
Reservations	High	15
Rents	High	15
Cars	Low	7
Safe Areas	Low	7
Charging Stations	Low	7
Total		75

2.1.2 External Interface Files (EIF)

The external interface files that the system relies on are the one coming from the external interfaces: the transport authority, the civil registry and the payment gateway. The system communicates also with an external interface to the operator system but since there are no input data from that interface we will not consider it. Here we will analyze the three EIFs:

- During the registration phase the user has to input his driving license number; this data is used to call an external service provided by the transport authority and to verify that the driving license is valid for our purpose. This is an operation that happens rarely (once per client) and that involves only one structure (client data) using only one request, so we can consider this EIF as a **Low** complexity one.

- During the registration phase the user has to input his ID number, this data is used to call an external service provided by the civil registry to verify the validity of the data inserted. As the one above this operation happens rarely (once per client) and involves only one structure (client data) using only one request, so we can consider this EIF as a **Low** complexity one.

- After that a client has ended the rent the system start the payment process throw the payment gateway. From this process we acquire the receipts for the payment that has to be stored. Since this operation will take place pretty often and since these are sensible data and for this reason have to be stored in an high security data structure we can consider this EIF as an **High** complexity one.

Summarizing this function we can derive the following table:

EIF	Complexity	FPs
Driving License	Low	5
Personal Data	Low	5
Payments Data	High	10
Total		20

2.1.3 External Inputs(EI)

The PowerEnjoy system has to interact with our clients for different purposes (In order to understand the complexity of these functions we will refer often to the components described in the components diagram of the design document):

- **Login/Logout:** This is an operation that requires only a component (UserController) and just with one entity, so we can classify this function as a **Low** complexity one.
- **Registration:** This operation requires one component and the interaction with two external interfaces so we can estimate its complexity as **Average**.
- **Reservation Request:** This operation requires the cooperation of a few components; we will have to verify if the car is available (CarController), if so we have to insert the reservation in the database, modify the status of the car (CarRemoteController and CarController) and send the feedback to the user so we can estimate its complexity as **Average**.
- **Cancel Reservation:** This operation requires the same components as the one before so reasoning in the same way we can estimate its complexity as **Average**.
- **Report Damaged Car:** This operation requires a complex function in the CarRemoteController that has to interact with the server system (using CarController) and then communicate with the operator system using an external interface, so given the number of components and the interaction throw an external interface we can estimate its complexity as **High**.
- **Start rent:** this operation requires the cooperation of quite all the components of our system and it has to query the database changing the status of the rent and the reservation, so we can estimate this operation as an **High** complexity function.

- **End rent:** This operation requires the cooperation of a lot of components: CarController for the car position, CarController to change the status of the car and check if the car is plugged, RentController to check information about the rent and others. It also has to start the payment process throw an external interface and it has to query the database to update the rent data. We can consider it an **High** complexity function.
- **Pin Insertion:** After that a client enters in a car and wants to start the engine he has to authenticate himself inserting the pin in the car system. This operation is just a simple check that requires the cooperation of few components(CarController, CarRemoteController) and a query in the database. It also has to use an high security protocol so we can say that the complexity of this function is **Average**

Summing up all the function spotted we derive the following table:

EI	Complexity	FPs
login/logout	Low	2x3
Registration	Average	4
Reservation Request	Average	4
Cancel Reservation	Average	4
Report Damaged Car	Average	6
Start rent	High	6
End rent	High	6
Pin Insertion	Average	4
Total		40

2.1.4 External Outputs(EO)

The PowerEnJoy system communicate tos the external world throw several streams and for several purposes, here below we have identified the most important:

- **Money saving option:** When the client selects the money saving option the system has to elaborate what is the best charging station based on the destination and the cars distribution. Once that the computation is done the data is sent to the CarRemoteController and so to the client throw the car display. The computation for this function is algorithmic hard and requires the cooperation of several components so we can classify its complexity as **Hard**.
- **Report damaged/dirty car:** As specified in the RASD document this function allows the user to report to the operator system issues concerning the car from the car display. To accomplish this function are necessary a couple of components and a interaction with an external interface so we can estimate the complexity of the function as **Average**.
- **Signal flat cars:** If the car system finds out that the battery is empty it starts the procedure of signaling the issue to the operator system. This function requires the cooperation of the CarRemoteController, of the CarController and the interaction with an external interface so we can estimate its complexity as **Average**.
- **Notify reservation time expired:** This is a simple operation that requires only few components to be performed, so we can estimate its complexity as **Low**.
- **Set car status:** This operation is simple because it requires the cooperation of only two components(CarController, CarCemoteController). We can say that its complexity is **Low**.
- **Display money charged:** For this function are necessary only a chronometer, a multiplication ad to display on the screen a number. All these operation are trivial and made by just one component so we can classify the function as a **Low** complexity one.
- **Send email with pin and password:** After that a client registers in the system it will receive an email with his pin and his password. To accomplish this function are necessary a password and pin generator and an automatic email generator. These are trivial tasks and we can say that the operation has a **Low** complexity.

The overall estimation for external outputs is summarized in the following table:

EO	Complexity	FPs
Money saving option	High	7
Report damaged/dirty car	Average	5
Signal flat cars	Average	5
Notify reservation time expired	Low	4
Set car status	Low	4
Display money charged	Low	4
Send email with pin and password	Low	4
Total		33

2.1.5 External Inquiries (EQ)

External inquiries are essential function to retrieve data for our clients. The main inquiries are the one performed in order to retrieve map information and the one to retrieve personal user data. In this paragraph will be summarized the most important:

- **Retrieving charging station information:** This function is used by the client in order to populate the map. It requires just few information from the database and are required few components to accomplish this task so we estimate the inquiry as a **Low** complexity.
- **Car information Retrieval:** This function requires the interaction of few components but the data to be exchanged are a few. It also may require to refresh of the car information making a remote call on the CarRemoteController. This lets us say that the complexity is **Average**.
- **Safe area position:** This function is used by the client app or web app in order to populate the map. It requires a simple query to be accomplished so the complexity estimation for this function is **Low**.
- **Retrieving User Information:** This function is used in the personal area of the client in the web app and the app. Its complexity is **Low** because it requires just a simple query to the database in order to be accomplished.

In the following table we summarize the estimation of external inquiries:

EQ	Complexity	FPs
Retrieving charging station information	Low	3
Car information Retrieval	Average	4
Safe area position	Low	3
Retrieving User Information	Low	3
Total		13

2.1.6 Overall estimation

The following table summarizes the results of our estimation :

Function type	FPs
Internal Logic Files	75
External Interface Files	20
External Inputs	40
External Outputs	33
External Inquiry	12
Total	180

Considering Java Enterprise Edition as development platform and disregarding the aspects concerning the implementation of the mobile applications (which can be thought as pure presentation with no business logic), we can estimate the total number of lines of code. Depending on the conversion rates, we have a lower bound of:

$$SLOC = 180 \cdot 46 = 8280$$

and an upper bound of :

$$SLOC = 180 \cdot 67 = 12060$$

2.2 Cost and effort estimation:COCOMO II

In this section we are going to use COCOMO II to provide a cost and effort estimation of the PowerEnJoy project. COCOMO II (COnstructive COst MOdel) is a complex estimation model that permits to estimate the number of Person/Month required to develop a complex project using an *Effort Equation* derived from fitting a regression formula using data from historical projects. We are going to use the *post-architecture* approach because we have already defined the architecture previously in the Design Document.

2.2.1 Scale Drivers

In order to compute the Effort Equation of the COCOMO II model we have to consider the scale drivers of our project. Below is provided the official COCOMO II scale driver table. The selection of scale drivers is based on the rationale that they are a significant source of exponential variation on a project's effort. Each driver has a range of rating levels, from Very Low to Extra High and each rating level has it's own scale factor.

Scale Factor values 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
SF_j	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
SF_j	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%)
SF_j	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
SF_j	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
SF_j	7.80	6.24	4.68	3.12	1.56	0.00

In order to obtain the scale factors we have chosen for each scale driver the rating value as explained below.

- Precedentedness (**PREC**)
  This driver reflects the experience that developers have with previously developed projects similar to PowerEnJoy. Since this is our first time managing such a complex project this value will be considered Very Low.
- Development flexibility (**FLEX**)
  This driver reflects the flexibility in the development with respect to pre-established requirements and external interface specifications. We have to conform to strict specifications on the functionalities and to interact with several external systems that have a well defined interface so this value will be considered Low.
- Risk resolution (**RESL**)
  This driver reflects the completeness and goodness of the risk management plan. This value will be considered High because we did a detailed Risk Analysis (Chapter 5).
- Team cohesion (**TEAM**)
  This driver reflects how much the team is cohesive. We are a very cooperative team and we share the same commitment so this value will be considered Very High.
- Process maturity (**PMAT**)
  This driver reflects the process maturity of the organization. In our case we will consider the Nominal value (CMM Level 2) due to our inexperience.

The summary of our evaluation is :

Scale Driver	Factor	Value
Precedentedness(PREC)	Very Low	6.20
Development flexibility (FLEX)	Low	4.05
Risk resolution (RESL)	High	2.83
Team cohesion (TEAM)	Very high	1.10
Process maturity (PMAT)	Nominal	4.68
Total		18.86

2.2.2 Cost Drivers

In this section we are going to describe which are the cost drivers that needs to be considered in the effort evaluation. Each driver has a range of rating levels, from Very Low to Extra High. For each driver is provided a table that maps the rating level with the effort multiplier.

- Required Software Reliability **(RELY)**:

  This driver reflects the expected software reliability. A downtime of our service would cause problems to passengers but would not lead to high financial losses. Therefore we will consider this value Nominal.

RELY Cost Drivers						
RELY Descriptors	slightly inconvenience	easily recoverable losses	moderate recoverable losses	high financial loss	risk to human life	
Rating level	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)**:

  This driver attempts to capture the effort required to generate the test data during the development. That's strictly related to the amount of data that will be stored in our system. We expect a large amount of data saved in our database due to the complexity of the service provided therefore this value will be considered High.

DATA Cost Drivers						
DATA Descriptors		Testing DB bytes/pgm SLOC < 10	10≤D/P≤100	100≤/P≤000	DP > 1000	
Rating level	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)**:

  We have chosen the rating scale of this cost driver according to the COCOMO II CPLX rating scale table. In this table Complexity is divided into five areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operation. The value that fits most our project is Very High.

CPLX Cost Driver						
Rating level	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)**:

  This cost driver reflects the effort needed to construct components intended for reuse on the current or future projects. As we are going to consider only this project is needed reusability "across the project",therefore this value will be set to Nominal.

RUSE Cost Driver						
RUSE Descriptors		None	Across project	Across program	Across product line	Across multiple product lines
Rating level	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)**:

  This cost driver reflects the suitability of the project's documentation to its life-cycle needs. As we have a detailed documentation that covers large part of the life-cycle needs of the project we are going to consider this value Nominal.

DOCU Cost Driver						
DOCU Descriptors	Many life-cycle needs uncovered	Some life-cycle needs uncovered	Right-sized to life-cycle needs	Excessive for life-cycle needs	Very excessive for life-cycle needs	
Rating level	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)**:

  This driver reflects the execution time constraint imposed upon the system. Every rating express the percentage of available execution time that is going to be used by the system. In our case we expect a Very High usage of the execution time because our system is large and complex.

TIME Cost Driver						
TIME Descriptors			$\leq$ 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 level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	n/a	1.00	1.11	1.29	1.63

- Main storage constraint **(STOR)**:

  This driver measures the degree of main storage constraint imposed on the system. In the market are available main storage disks large enough for our system. Therefore this value will be considered Nominal.

STOR Cost Driver						
STOR Descriptors			$\leq$ 50% use of available storage	70% use of available storage	85% use of available storage	95% use of available storage
Rating level	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)**:

  This driver reflects how frequently a major change of the platform is required. The "platform" is the complex of hardware and software required by the system to perform the tasks. We don't expect frequent changes in our main platform but we forecast to release every six months updates of the client application in order to follow smartphones operatives system progression. According to the table below we will consider this value Nominal.

PVOL Cost Driver						
PVOL Descriptors		Major change every 12 mo., minor change every 1 mo.	Major: 6mo; minor: 2wk.	Major: 2mo, minor: 1wk	Major: 2wk; minor: 2 days	
Rating level	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)**:

  The analysis and design of this project have been done thoroughly and precisely. Therefore this value should be set to High.

ACAP Cost Driver						
ACAP Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating level	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)**:

  This driver cost should consider the capability of the programmers as a team rather than as individuals. Even if we haven't implemented the project, we value the ability to communicate and cooperate of our team. Therefore we will consider this value as Nominal.

PCAP Cost Driver						
PCAP Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.34	1.15	1.00	0.88	0.76	n/a

- Application Experience **(APEX)**:

  The rating of this driver is chose in terms of the team's level of experience with this type of software application. Our experience related to this type of application is very low so this value will be considered Very Low.

APEX Cost Driver						
APEX Descriptors	≤ 2 months	6 months	1 year	3 years	6 years	
Rating level	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)**:

  Our experience with platforms as graphic user interface, database, networking and Client/Server architecture consist of 1 year. According to the table below we will consider this value Nominal.

PLEX Cost Driver						
PLEX Descriptors	≤ 2 months	6 months	1 year	3 years	6 years	
Rating level	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)**:

  This driver measures the level of programming language and software tool experience of the project team. Our average experience with Java is almost 1 year, we also have some background on HTML but we don't have any practical experience with JEE. This value will be considered Low.

LTEX Cost Driver						
LTEX Descriptors	≤ 2 months	6 months	1 year	3 years	6 years	
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.20	1.09	1.00	0.91	0.84	n/a

- Personnel continuity **(PCON)**:

  This driver reflects the project's annual personnel turnover. Even if we are not going to develop the project we can expect a low turnover because we suppose to work together toward the project completion. This value will be considered Very High

PCON Cost Driver						
PCON Descriptors	48% / year	24% / year	12% / year	6% / year	3% / year	
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.29	1.12	1.00	0.90	0.81	n/a

- Usage of Software Tools **(TOOL)**:

  We are going to use basic commercial tools so this value will be considered Nominal.

TOOL Cost Driver						
TOOL Descriptors	edit, code, debug	simple, frontend, backend CASE, little integration	basic life-cycle tools, moderately integrated	strong, mature life-cycle tools, moderately integrated	strong, mature, proactive life-cycle tools, well integrated with processes, methods, reuse	
Rating level	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)**:

  This driver reflects two factors : site collocation and communication support. All the members of our team live in the same city. We also use chats, emails and phone calls to communicate. For this reason we are going to consider this value Very High.

SITE Cost Driver						
SITE Collocation Descriptors	International	Multi-city and multi-company	Multi-city or multi-company	Same city or metro area	Same building or complex	Fully collocated
SITE Communications Descriptors	Some phone, mail	Individual phone, fax	Narrow band email	Wideband electronic communication	Wideband elect. comm., occasional video conf.	Interactive multimedia
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.22	1.09	1.00	0.93	0.86	0.80

- Required development schedule **(SCED)**:

  We don't have any special schedule constraint imposed for the software developing so this value will be considered Nominal.

SCED Cost Driver						
SCED Descriptors	75% of nominal	85% of nominal	100% of nominal	130% of nominal	160% of nominal	
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.43	1.14	1.00	1.00	1.00	n/a

Overall, our results are expressed by the following table:

Cost Driver	Factor	Value
Required Software Reliability (RELY)	Nominal	1.00
Database size (DATA)	High	1.14
Product complexity (CPLX)	Very high	1.34
Required Reusability (RUSE)	Nominal	1.00
Documentation match to life-cycle needs (DOCU)	Nominal	1.00
Execution Time Constraint (TIME)	Very high	1.29
Main storage constraint (STOR)	Nominal	1.00
Platform volatility (PVOL)	Nominal	1.00
Analyst capability (ACAP)	High	0.85
Programmer capability (PCAP)	Nominal	1.00
Application Experience (APEX)	Very Low	1.22
Platform Experience (PLEX)	Nominal	1.00
Language and Tool Experience (LTEX)	Low	1.09
Personnel continuity (PCON)	Very High	0.81
Usage of Software Tools (TOOL)	Nominal	1.00
Multisite development (SITE)	Very high	0.86
Required development schedule (SCED)	Nominal	1.00
Product		1,5516

2.2.3 Effort equation

Now that we have considered both scale drivers and cost drivers we can estimate the number of Person/Month required to develop the project. In order to do so we compute the following Effort Equation :

$$Effort = A * EAF * KSLOC^{E}$$

$A = 2.94$ approximates a productivity constant in PM/KSLOC.
EAF is the product of all cost drivers. In our case it is $EAF = 1.5516$.
Using function points estimation we have estimated :

lower bound of $KSLOC$

$$KSLOC = 8.280$$

upper bound of $KSLOC$

$$KSLOC = 12.060$$

The exponent E is derived from the aggregation of the five Scale Drivers. It is calculated as:

$$E = B + 0.01 * \sum_{j=1}^{5} SF_j$$

Where $B = 0.91$ is an empirically proven constant in COCOMO II.
We obtain that :

$$E = 0.91 + 0.01 * 18.86 = 0.91 + 0.1886 \cong 1.0986.$$

Finally we can compute the effort needed:

lower bound of $Effort$

$$Effort = 2.94 * 1.5516 * 8.280^{1.0986} = 46.52\ PM$$

upper bound of $Effort$

$$Effort = 2.94 * 1.5516 * 12.060^{1.0986} = 70.32\ PM$$

2.2.4 Schedule estimation

We can estimate the project duration using the following equation:

$$Duration = 3.67 * Effort^{F}$$

$Effort$ is the previously estimated effort and F is the schedule exponent derived from the Scale Drivers. We calculate F using the following formula:

$$F = D + 0.2 * (E - B)$$

that is equivalent to write

$$F = D + 0.2 * 0.01 * \sum_{j=1}^{5} SF_j = 0.28 + 0.2 * 0.01 * 18.86 \cong 0.3177$$

Now we can calculate:

lower bound of $Duration$

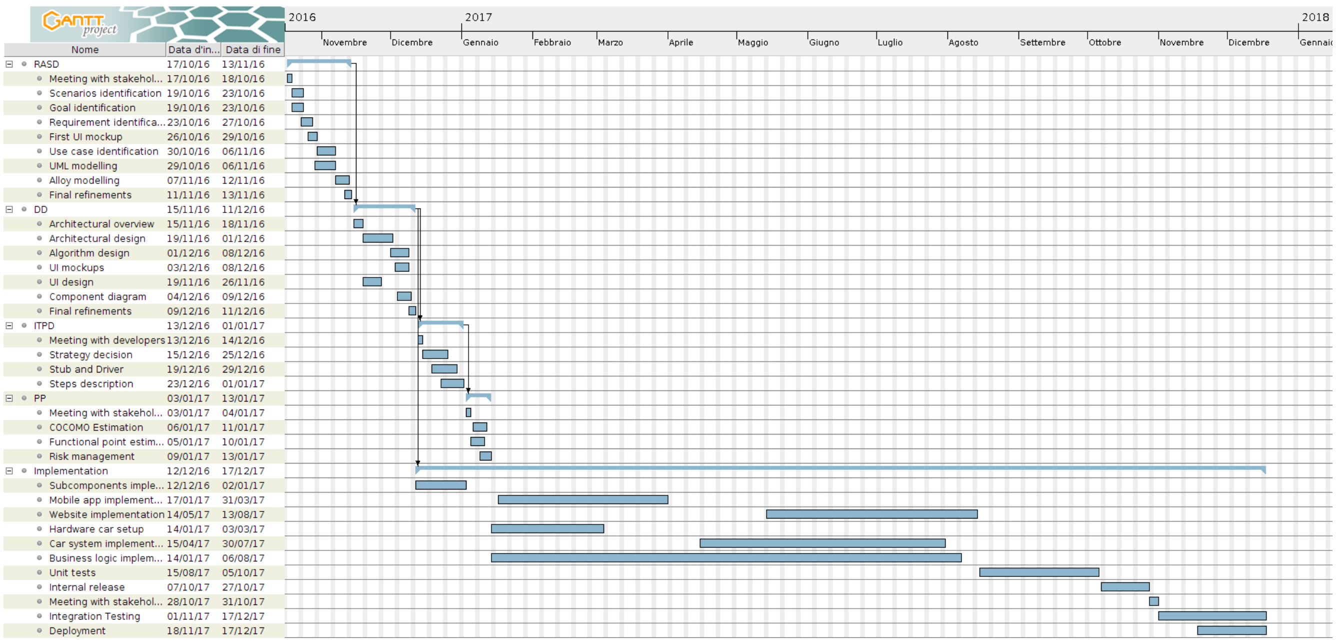
$$Duration = 3.67 * 46.52^{F} = 3.67 * 46.52^{0.3177} \cong 12.43\ Months$$

upper bound of $Duration$

$$Duration = 3.67 * 70.32^{F} = 3.67 * 70.32^{0.3177} \cong 14.17\ Months$$

3 Schedule

In order to accomplish this project, we have to follow some steps. First of all we should write the **Requirement Analysis and Specification Document**. This document helps us to focus on functional and non-functional requirements, domain assumption and goals of the our project. Thus, we can write the **Design Document** where we will define the architecture and the component that will be developed. Then, we start to code the **Implementation** and in parallel, we write the **Integration Testing Plan Document**. Here we explain the strategy we want adopt to test the integration between the system components. The last document to be released is the **Project Plan**, that is this document. After that, we prepare a briefly **Presentation** of the main part of the previous mentioned document. Finally, when the implementation is almost finished, the **Integration Test** will be performed.
The schedule for each document should observe some deadline definite by our customer. Instead, the implementation phase doesn't have a particular deadline, so we estimate its duration in 13 months, like calculated in section 2. In order to better explain the schedule of the project, we provide a Gantt diagram here below.

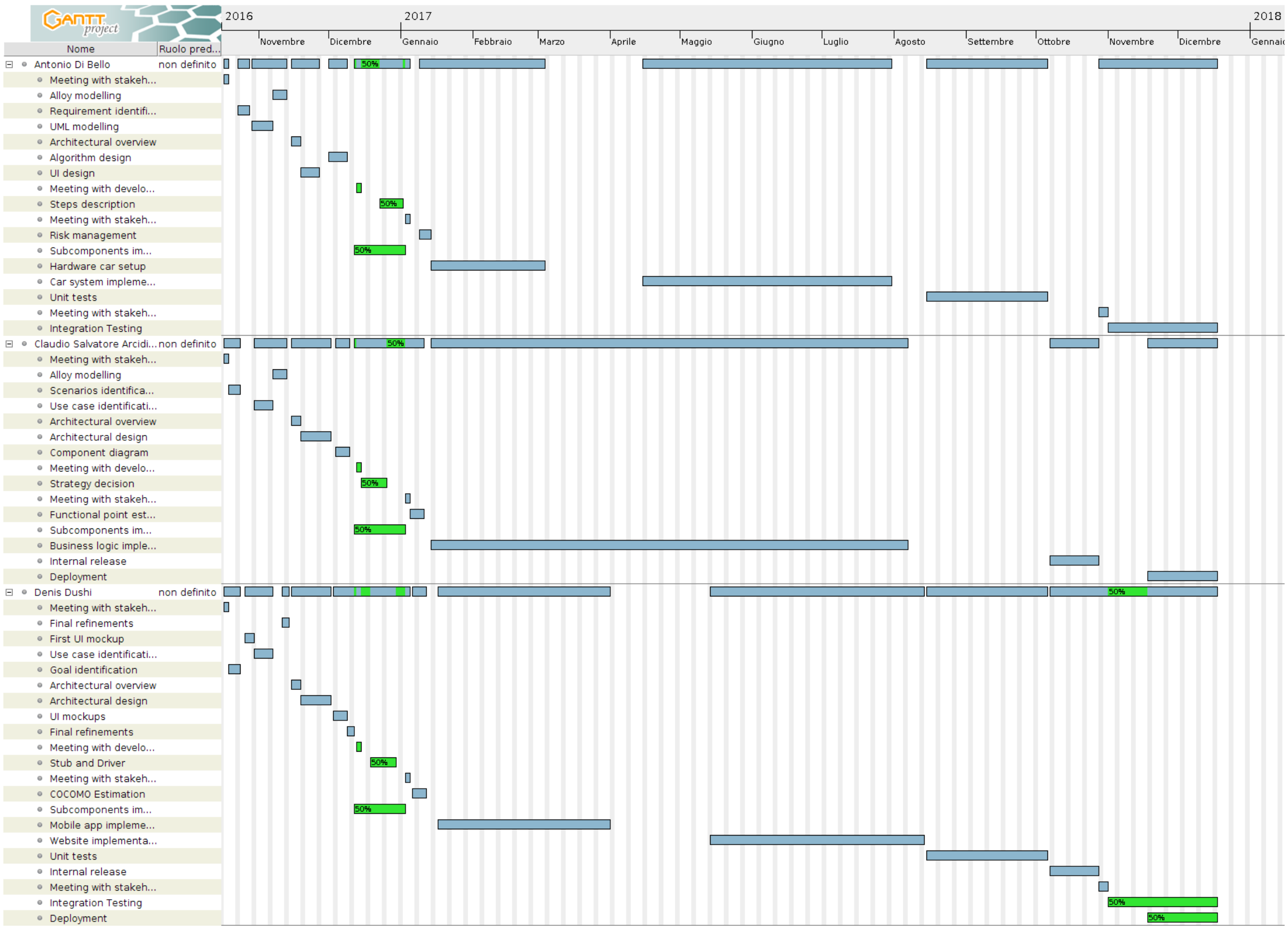


Nome	Data d'in...	Data di fine
RASD	17/10/16	13/11/16
Meeting with stakehol...	17/10/16	18/10/16
Scenarios identification	19/10/16	23/10/16
Goal identification	19/10/16	23/10/16
Requirement identifica...	23/10/16	27/10/16
First UI mockup	26/10/16	29/10/16
Use case identification	30/10/16	06/11/16
UML modelling	29/10/16	06/11/16
Alloy modelling	07/11/16	12/11/16
Final refinements	11/11/16	13/11/16
DD	15/11/16	11/12/16
Architectural overview	15/11/16	18/11/16
Architectural design	19/11/16	01/12/16
Algorithm design	01/12/16	08/12/16
UI mockups	03/12/16	08/12/16
UI design	19/11/16	26/11/16
Component diagram	04/12/16	09/12/16
Final refinements	09/12/16	11/12/16
ITPD	13/12/16	01/01/17
Meeting with developers	13/12/16	14/12/16
Strategy decision	15/12/16	25/12/16
Stub and Driver	19/12/16	29/12/16
Steps description	23/12/16	01/01/17
PP	03/01/17	13/01/17
Meeting with stakehol...	03/01/17	04/01/17
COCOMO Estimation	06/01/17	11/01/17
Functional point estim...	05/01/17	10/01/17
Risk management	09/01/17	13/01/17
Implementation	12/12/16	17/12/17
Subcomponents imple...	12/12/16	02/01/17
Mobile app implement...	17/01/17	31/03/17
Website implementation	14/05/17	13/08/17
Hardware car setup	14/01/17	03/03/17
Car system implement...	15/04/17	30/07/17
Business logic implem...	14/01/17	06/08/17
Unit tests	15/08/17	05/10/17
Internal release	07/10/17	27/10/17
Meeting with stakehol...	28/10/17	31/10/17
Integration Testing	01/11/17	17/12/17
Deployment	18/11/17	17/12/17

Figure 1: Gantt diagram for PowerEnJoy project.

4 Resource Allocation

Here we will go to define how the tasks are allocated between the three members of the team. Since our team is composed by homogeneous people, we can consider the same amount of time for each person.



Nome	Ruolo pred...
Antonio Di Bello	non definito
Meeting with stakeh...	
Alloy modelling	
Requirement identifi...	
UML modelling	
Architectural overview	
Algorithm design	
UI design	
Meeting with develo...	
Steps description	
Meeting with stakeh...	
Risk management	
Subcomponents im...	
Hardware car setup	
Car system impleme...	
Unit tests	
Meeting with stakeh...	
Integration Testing	
Claudio Salvatore Arcidi...	non definito
Meeting with stakeh...	
Alloy modelling	
Scenarios identifica...	
Use case identificati...	
Architectural overview	
Architectural design	
Component diagram	
Meeting with develo...	
Strategy decision	
Meeting with stakeh...	
Functional point est...	
Subcomponents im...	
Business logic imple...	
Internal release	
Deployment	
Denis Dushi	non definito
Meeting with stakeh...	
Final refinements	
First UI mockup	
Use case identificati...	
Goal identification	
Architectural overview	
Architectural design	
UI mockups	
Final refinements	
Meeting with develo...	
Stub and Driver	
Meeting with stakeh...	
COCOMO Estimation	
Subcomponents im...	
Mobile app impleme...	
Website implementa...	
Unit tests	
Internal release	
Meeting with stakeh...	
Integration Testing	
Deployment	

Figure 2: Resource allocation for each task.

5 Risks management

In this section we are going to explain which are the risks that could affect the realization of our project. Threats could derive from technical issues, from a wrong business analysis or could be related to controversies with our stakeholders. For each risk we are going to estimate the probability of occurrence and we will provide a possible contingency plan.
The first risk that could jeopardize the project comes from aggressive competition. The competitors that provide a car-sharing service are few but consolidated, therefore they could try to initiating a price war in order to discourage new entrants. The probability that this occurs is high and, without the right precautions, this would affect enormously our business. This risk can be mitigated with a good marketing strategy, in particular is necessary to promote our competitive advantage (electric cars and eco-friendly behaviors) in order to differentiate from competitors.
Another potential issue could come from the lack of free space where to build the charging stations or from social movements that protest against the realization of charging stations in cultural areas. In both the cases to overcome the problem we have to make clear to the city administration the advantages that PowerEnJoy provides to the local community and to let the stakeholders have an active role in the development of the project through periodical meetings and demonstrations.
Other issues that could be critical have an internal origin : key members of the development team could be ill in periods relevant for the project realization or they could just quit the company. This could lead to delays and a lack of knowledge related to a part of the project. To avoid so it's important to maintain a high quality documentation and to organize the team work assuring that duties and responsibilities are assigned across multiple team members so no critical task is assigned to a single person.
We have also to consider the possibility that we underestimate the development time necessary to completely develop the project. In this case allocating new budged in order to add manpower to the project is not recommended, if done in a late phase of development would increase even more the delay! Instead is fundamental to promptly alert the customer of the possible delay and possibly accord on a two-phase release in order to provide as soon as possible a working system and later to add and refine functionalities.
Other risks might arise from technical issues that could threat the quality of the product and increase the costs of the software developing process. The application might have security issues if not well designed. This could lead to a leak of users' sensible data that would be catastrophic for PowerEnJoy's image and therefore for the business of the company. In this case we have to follow all the security standards and test very well the most critical parts to lower as much as possible the probability of occurrence.
Other risks that should be considered are related to our dependency on external components. The first suppliers of PowerEnJoy are electric cars vendors. The cars must have a centralized system to provide the possibility to lock/unlock doors remotely. Vendors could try to set inconvenient pricing plans for this additional feature. In this case, since PowerEnJoy needs a large amount of cars, we should leverage on the economy of scale to have lower prices. Other components that have to be acquired externally are the car displays but this shouldn't be an issue due to the large presence of similar devices in the market.

6 Effort spent

During the whole project the team worked with the following schedule:

Claudio Salvatore Arcidiacono

- 17 January : 16.30-18.30 2 hours
- 18 January : 15-20 5 hours
- 19 January : 17-21 4 hours
- 20 January : 10-13 3 hours
- 21 January : 17-20 3 hours

Total hours: 17 hours.

Antonio Di Bello

- 20 January : 1 hours
- 21 January : 6 hours
- 22 January : 3 hours

Total hours: 10 hours.

Denis Dushi

- 16 January : 2 hours
- 17 January : 2 hours
- 19 January : 7 hours
- 20 January : 6 hours
- 21 January : 5 hours
- 22 January : 1 hour

Total hours: 23 hours.

7 References

7.1 Software and tool used

The tools we used to create this document are:

- Overleaf (for latex writing in parallel)