PowerEnJoy
Project Plan Document
Version 1

Patricia Abbud
Maddalena Andreoli Andreoni
Paolo Cudrano

Contents

1 Introduction

1.1 Purpose and scope

The present document is the **Project Plan Document** for the *PowerEnJoy* system. It aims to analyse the dimension of the project and the effort required, to guide and assist the management of all the involved processes.
In **section 2**, a post-architecture analysis of the dimension and effort cost of the project is performed. An estimate of the source lines of code (SLOC) is computed applying the Function Point (FP) approach. Subsequently, this evaluation is used to predict the actual effort the project will require, using the COCOMO II framework. The result can be used as a reference parameter to update the budget evaluation and provide an estimate of the resource allocation.
Section 3 and **section 4** take into account the effort estimation to provide a possible schedule and a human resource allocation for the project.
Finally, **section 5** conducts an analysis of the most critical risks the project may face, trying to describe the preventive contingency plans to be adopted.

1.2 Definitions, acronyms, abbreviations

Provided here is a short list of acronyms often used inside this document. Please refer to the previous documents' chapter on definitions and abbreviations for any term not described here.

1.2.1 Acronyms

- **SLOC**: Source Lines Of Code, a software metric used to measure the size of a system.
- **FP**: Function Points, a size estimation method for software.
- **COCOMO**: COnstructive COst MOdel, a software cost estimation model. In this document, the evolution COCOMO II is used.

1.3 Reference documents

Here is provided a list of documents we used as reference.

PowerEnJoy RASD : the Requirement Analysis and Specification Document for *PowerEnJoy*'s system.

PowerEnJoy DD : the Design Document for *PowerEnJoy*'s system.

Project Assignment : the project assignment document, *Assignments AA 2016-2017.pdf.*

`http://www.softwaremetrics.com/fpafund.htm` A FP guide.

`http://www.qsm.com/resources/function-point-languages-table` The QSM Function Points Languages Table, containing the function point language gearing factors.

`http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf`
The COCOMO II Model Definition Manual.

2 Project Size, Cost and Effort Estimation

This section is intended to provide an estimation of the expected size, cost and effort required by *PowerEnJoy*. We will exploit algorithmic techniques to make these estimations; we will use Function Points to appraise the size of the future system, taking into account only the business logic and its functionalities, without taking into consideration the user view. For the cost and effort estimation we will use the COCOMO approach, using the SLOC result from the FP approach as a starting point for its calculations.
Both the size and the cost and effort estimations will be done post-architecture: the model thus created will be detailed and pertains a project which is ready to develop, as is ours (since we have all the necessary documents already written).

2.1 Size estimation: Function points

The Function Points approach provides an estimation of the source lines of code (i.e. of the dimension of the project) based on the functionalities that need to be developed and their complexity. This estimation is made as objective as possible by a series of tables that provide normalized values for different levels of complexity. However, it is necessary to remember that the final choice of complexity belongs to the analyst, that may underestimate or overestimate the effort required for a particular functionality. The COCOMO estimation will also account for the degree of error the analyst will be subject to.

The following tables are those we reference while choosing values for our function points:

Record Element Types for ILFs and ELFs:

RET's	**DATA ELEMENTS**		
	1-19	20-50	>50
1	Low	Low	Avg
2-5	Low	Avg	High
>5	Avg	High	High

Function points for ILFs and ELFs:

Rating	**VALUES**	
	ILFs	ELFs
Low	7	5
Avg	10	7
High	15	10

Record Element Types for EOs, EIQs, EIs:

RET's	**DATA ELEMENTS**		
	1-5	6-19	>19
0-1	Low	Low	Avg
2-3	Low	Avg	High
>3	Avg	High	High

Function points for EOs, EIQs, EIs:

Rating	**VALUES**		
	EO	EIQ	EI
Low	4	3	3
Avg	5	4	4
High	7	6	6

2.1.1 Internal Logic Files (ILFs)

The system of *PowerEnJoy* will require a series of ILFs to provide the required functionalities. In this section we will quickly list them and then offer an overall evaluation.

- **Users**: the system needs to store information about all users in order to correctly function. These data all belong to a set of three tables containing their payment information (credit card number, expiration date, CVV), driving license information (driving license number, expiration date, issued date) and account information (full name, username, password, telephone number, email).
- **Parking areas**: we need to store all the information about parking areas. In particular, whether they are recharging areas, how many slots they have and their location. The system will also have to know the number of available slots at a given time.
- **Cars**: the system needs to store information about its cars, most notably their plate number, their location and their status.
- **Rides**: the system needs to keep track of the rides; these data belong to a table that will need also references to a car and a user, and will have a duration and applied discounts or sanctions.
- **Reservations**: the system also needs to keep track of the reservations; those will contain a reference to a car and a user as well, and they will have a time period.
- **Operators and Admins**: the system needs to store information also about the back-end users, i.e. the operators and the admins: those are characterized with a username, a password, their full name and email address, and their role.
- **Emergency Reports**: for the system to work even in case of emergencies, it needs to track emergency reports. They are linked to a particular ride (and a car and a user through it) and an Operator.

- **Parameters**: the admins must be able to modify some of the parameters of the system; those parameters will have a name, a description and a value.
- **Fees**: all fees must be linked to a user and a particular ride.

Using the previosly defined tables, we obtain the following FP count:

ILF	**Complexity**	**FPs**
Users	Average	10
Parking Areas	Average	10
Cars	Low	7
Rides	High	15
Reservations	High	15
Operators/Admins	Low	7
Emergency Reports	Average	10
Parameters	Low	7
Fees	Average	10
Total:		91

2.1.2 External Interface Files (ELFs)

PowerEnJoy relies on one external data source, that is the Location Services Provider. There are a series of interactions between the system and the Location Services Provider:

- Given an address, return the coordinates of said address;
- Given a user's GPS, return the coordinates of their position;
- Given two locations, return the approximate distance between the two;
- Retrieve the graphical representation of the map on the user's phone (i.e. on a client).

We thus obtain the following FP count:

ELF	**Complexity**	**FPs**
Distance computation	High	10
Geolocation	High	10
Map retrieval	Average	7
Total:		27

2.1.3 External Input (EIs)

This section will recite and evaluate the interactions between *PowerEnJoy*'s system and users.

All users:

- **Login/Logout**: these are farily simple operations, so we can adopt the simple weight for both of them: it contributes $2 \times 3 = 6FPs$.

Front-end users (and guests):

- **Register**: this operation is fairly complex as it requires a lot of external input to be submitted: it contributes $6FPs$.
- **Change account settings**: this operation is elementary, hence we apply the simple weight for it: it contributes $3FPs$.
- **Reserve a car**: this is a very complex operation that involves a large number of components: it contributes $6FPs$.

Operators:

- **Change emergency report status**: this is an average operation since the status of an emergency report has consequences for the car and the ride, hence it contributes $4FPs$.

Administrators:

- **Change system parameters**: this is an average operation, because it involves a number of checks in the field validity. So it contributes $4FPs$.

So, to summarize:

EI	**Complexity**	**FPs**
Login/Logout	Low	$2 \times 3 = 6$
Register	High	6
Change account settings	Low	3
Reserve a car	High	6
Change Emergency report status	Average	4
Change system parameters	Agerage	4
Total:		29

2.1.4 External Inquiries (EIQs)

PowerEnJoy supports many possible external inquiries performed by various users:

Users:

- Retrieve information about available cars;
- Retrieve information about parking areas nearby;

- Retrieve information about their ride;
- See their profile.

Operators:

- Retrieve information about a selected car;
- Retrieve information about an emergency report;
- See their profile.

Administrators:

- Retrieve information about a selected car;
- Retrieve information about operators;
- Retrieve information about parameters;

Here is how we evaluated the complexity of each of these functionalities:

EIQ	**Complexity**	**FPs**
Available cars	High	6
Parking areas	High	6
Ride	Average	4
Profile	Low	3
Selected car	Average	4
Emergency Report	Average	4
Operators	Average	4
Parameters	Low	3
Total:		34

2.1.5 External Outputs (EOs)

The system is also expected to communicate with the user even outside inquiries. There are a few occasions in which this happens, most notably:

1. Notify the user when a payment has been invoiced (normal fees or fines) and how much;
2. Notify the user that he is driving out of city boundaries;
3. Notify the administrators that a car requires assistance;
4. Nofity the operators that they have been dispatched.

While the last three are operations of average complexity, the first one is fairly complex because it requires interactions also with the Payment Services Provider. So, to make the sum:

EOs	**Complexity**	**FPs**
Payment notification	High	7
Out of boundary notification	Average	5
Assistance notification	Average	5
Dispatch notification	Average	5
Total:		22

2.1.6 Overall estimate: Unadjusted and adjusted function points

The following table summarizes the sum of the so-called UFP, i.e. the unadjusted function points:

Function Type	**Value**
Internal Logic Files	91
External Logic Files	27
External Inputs	29
External Inquiries	34
External Outputs	22
Total	203FPs

We can estimate the number of lines of code by taking into account that our system will be developed using JEE and disregarding for the sake of simplicity the client's implementation (mobile application), which can be considered as presentation with a modicum of business logic.
Using the J2EE conversion rates, we obtain an average of:

$SLOC_{avg} = 203 \times 46 = 9338$

And a higher bound of:

$SLOC_{high} = 203 \times 67 = 13601$

2.2 Cost and Effort estimation: COCOMO

In this section we will use the COCOMO II approach in order to estimate the cost and effort required by *PowerEnJoy*'s development.

We will use the results drawn in subsection 2.1 and the estimated Source Lines of Code to use as values in COCOMO II equations. To decide on the values of scale factors and cost drivers, we will refer to the official COCOMO II manual (see subsection 1.3 for hyperlink and further detail). The reference tables will be re-drawn for readability in each section, referencing them as *"CII modelman"*.

The following equations will be used to calculate the effort (in persons month) and duration (in months) of our project:

$Effort = A \times EAF \times KSLOC^E$
wherein

$E = B + 0.01 \times \sum_{j=1}^{5} SF_j$
is the exponent (B is a constant that in COCOMO II has value $B = 0.91$), A is a constant that in COCOMO II has value $A = 2.94$, and

$EAF = \prod_{i=1}^{17} CD_i$.

For the duration, we will use the following formula:

$Duration = C \times Effort^F$
where $C = 3.67$ and

$F = D + 0.2 \times (E - B) = 0.28 + 0.2 \times (E - 0.91)$.

2.2.1 Scale Factors

Scale factors (SF) are used to "account for the relative economies or diseconomies of scale encountered for software projects of different sizes" [Banker et al. 1994]. They are used to calculate the exponent E and there are five of them:

Scale Factor values, SF_j

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

Here is an explanation of each Scale Factor and our choices:

- **PREC**: Precedentedness describes "if a product is similar to several previously developed projects"[1]. While on the one hand we have no experience with this kind of system, it is also true that we won't need particular innovation and that similar system have been developed before. For these reasons, this factor is set to **Nominal**.

- **FLEX**: Flexibility describes the need for the software to conform to pre-established requirements and law. This factor is set to **Very Low**, meaning we have close to no flexibility in regards of either our requirements or state law regarding car driving and payments. Some (very) occasional relaxation may occur in the development of the company requirements.

- **RESL**: Architecture/Risk Resolution describes our awareness and preparedness with respect to risks; in particular it focuses on architecture and how much interest it has been given to its design and its uncertainties. We believe we have analysed as best as possible the risks inherent to our architecture, hence the value is set to **High**.

- **TEAM**: Team Cohesion accounts for "the sources of project turbulence and entropy because of difficulties in synchronizing the project's stakeholders"[2]. We believe our

[1]CII modelman, page 18
[2]CII modelman, page 20

team is cohesive, although it presents a language barrier problem. So this factor is set to **Very High**.

- **PMAT**: We believe that our inexperience in this kind of project allows for a pretty low project maturity. On the other hand we had only little problems during the planning stages of the project. To account for both those aspects, we set the value of this factor to **Low**.

To sum up our decisions, here is the following summary table:

Scale Driver	**Factor**	**Value**
PREC	Nominal	3.72
FLEX	Very Low	5.07
RESL	High	2.83
TEAM	Very High	1.10
PMAT	Low	6.24
Total:		18.96

2.2.2 Post-Architecture Cost Drivers

There are seventeen post-architecture cost drivers, which are multipliers used in COCOMO II in order to adjust the "nominal" effort in person-months in order to take into consideration the software under development. Nominal effort is defined as the effort estimated only using scale factors.

Product Factors

Product factors are needed to account for the variations in the effort required caused by the characteristics of the software under development. There are five of them:

- **RELY**: RELY measures the "extent to which the software must perform its intended function over a period of time"[3]. We decided to set it to **High** in order to keep into consideration the potential finantial losses caused by an accident in the car (even though the car itself is not driven by the system, a failure in notifying some problems with it may have undesired effects) and consequent risks to the driver.

RELY Descriptors	slight inconvenience	low, easily recoverable losses	moderate, easily recoverable losses	high financial loss	risk to human life	
Rating levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	0.82	0.92	1.00	1.10	1.26	n/a

[3]CII modelman, page 25

- **DATA**: this cost driver tries to "capture the effect large test data requirements have on product development"[4]. It is determined by calculating D/P, i.e. the size of the test database in bytes on the SLOC of the program. We have estimated our test database to be in the order of magnitude of 10 megabytes ($10^7 bytes$). Even if we count our SLOC with the average estimation of $\sim 9KSLOC$, the result is greater than 1000. Hence, the value of the cost driver DATA for us is **Very High**.

DATA Descriptors		$D/P < 10$	$10 \leq D/P < 100$	$100 \leq D/P < 1000$	$DP \geq 1000$	
Rating levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	0.90	1.00	1.14	1.28	n/a

- **CPLX**: Product Complexity is evaluated based on 5 factors, which are control operations, computational operations, device-dependent operations, data management operations, user interface management operations. Since we expect to have fairly complex operations concerning computation, devices, data management and user interface management, the complexity is set to **High**.

Rating levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	0.73	0.87	1.00	1.17	1.34	1.74

- **RUSE**: RUSE means Developed for Reusability. This cost driver was set to **Nominal** since we probably won't need the additional effort required to develop components for reusability outside of the project itself.

RUSE Descriptors		none	across project	across program	across product line	across multiple product lines
Rating levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	0.95	1.00	1.07	1.15	1.24

[4]CII modelman, page 26

- **DOCU**: the DOCU cost driver accounts for the dimensioning of the documentation with respect to the life-cycle needs. This driver is set to very low if the documentation is next to none and very high if the documentation is excessive. We set this value to **Nominal**.

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 levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	0.81	0.91	1.00	1.11	1.23	n/a

Platform Factors

Platform Factors take in consideration the complexity of hardware and infrastructure software needed for the system.

- **TIME**: the TIME factor measures the Execution Time Constraint imposed upon a software system. Since *PowerEnJoy* is quite a complex system, we expect a high execution time needed; on the other hand, our intention is to choose the dimentions of the CPU so that our system will use 85% of available execution time at operating time. For this reason the value is set to **Very High**.

TIME Descriptors			≤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 levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	n/a	1.00	1.11	1.29	1.63

- **STOR**: it represents the degree of Main Storage Constraint imposed on a system. Having a relatively small test database and being commercially available many storage units that can contain Terabytes of data, we don't believe storage will be a serious matter for us. Hence, this driver is set to **Nominal**.

STOR Descriptors			$\leq$50% use of available storage	70% use of available storage	85% use of available storage	95% use of available storage
Rating levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	n/a	1.00	1.05	1.17	1.46

- **PVOL**: The PVOL accounts for the volatility of the platforms on which the system is deployed. We believe that the clients operating systems may perform major updates than once a year; it is also true that we don't expect our other platforms to change often. For these reasons, we decided to set the PVOL value to **Nominal**.

PVOL Descriptors		Major change every 12 mo.; minor change every 1 mo.	Major: 6 mo; minor: 2 wk.	Major: 2 mo; minor: 1 wk.	Major: 2 wk.; minor: 2 days	
Rating levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	0.87	1.00	1.15	1.30	n/a

Personnel Factors

These cost drivers try to consider the personnel's abilities, experience and skills to put into the effort estimation. These are some of the most influential cost drivers, since the team's productivity and experience considerably changes the effort required to develop a project.

- **ACAP**: ACAP evaluates the capabilities of analysts, i.e. people working on requirements and design. The attributes to consider while evaluating ACAP are thoroughness, efficiency, ability, and the will to communicate and cooperate. We believe our team has worked well while drawing up requirements and deciding the design for the system, so we set this factor to **High**.

ACAP Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.42	1.19	1.00	0.85	0.71	n/a

- **PCAP**: PCAP evaluates the capabilities of the programmers as a team. The experience of the programmers is not rated here, since that will be evaluated later; factors to be considered during the evaluation are instead ability, thoroughness, efficiency and cooperation. Our team is cohesive and hard-working, but we believe that, since we're students, we may be lacking some knowledge and abilities. Hence this value is set to **Nominal**.

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

- **PCON**: Personnel Continuity stands for the annual turnover of personnel working on the project. We set it to **Very High** since we don't expect we will ask new people to help us or that some of us will leave the project.

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

- **APEX**: this cost driver evaluates the experience that the project team has with similar types of applications. We've never worked on similar projects before, neither with regards to the domain (i.e. car rentals) nor with its scope and dimentions; on the other hand we've been acquainting ourselves with the project system itself for the past few months, so we've decided to set this value to **Low**.

APEX Descriptors	≤ 2 months	6 months	1 year	3 years	6 years	
Rating levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.22	1.10	1.00	0.88	0.81	n/a

- **PLEX**: PLEX evaluates the same thing as APEX, but with respect to the platforms instead of the application. The team has been acquainting itself with the clients operating system, the cars' management system and so on for the duration of the project, and for this reason we set this cost driver to **Low**.

PLEX Descriptors	≤ 2 months	6 months	1 year	3 years	6 years	
Rating levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.19	1.09	1.00	0.91	0.85	n/a

- **LTEX**: LTEX evaluate the languages and tools experience the team has. We are all fairly acquainted with Java and DBMS languages; however we'd never worked with JEE before, nor with the tools related to it, so we set this value to **Nominal**.

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

Project Factors

This set of cost drivers account for the use of modern software tools, location of the development team, compression of the project schedule.

- **TOOL**: the TOOL rating changes based on the complexity, efficiency and timeliness of the development tools used. We have been using the most updated tools available open source, so this value is set to **Very High**.

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 levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.17	1.09	1.00	0.90	0.78	n/a

- **SITE**: SITE accounts on the collocation and communication support between teams (or team members, in our case). We all live in the same city and use occasional audio conferences and in general wide band electronic communication, so we set this driver to **High**.

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 electronic communication, occasional video conf.	Interactive multimedia
Rating levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.22	1.09	1.00	0.93	0.86	0.80

- **SCED**: this rating "measures the schedule constraint imposed on the project team developing the software"[5]. This value is set to **High**, because while we tried as much as possible to distribute the developing process, it is also true that the team members had other obligations with the university. So the compression of the schedule was high.

[5]CII modelman, page 34

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	Extra High
Effort multipliers	1.43	1.14	1.00	1.00	1.00	n/a

Our results are expressed in the summary table below:

Scale Driver	**Factor**	**Value**
RELY	High	1.10
DATA	Very High	1.28
COMPLEXITY	High	1.17
RUSE	Nominal	1.00
DOCU	Nominal	1.00
TIME	Very High	1.29
STOR	Nominal	1.00
PVOL	Nominal	1.00
ACAP	High	0.85
PCAP	Nominal	1.00
PCON	Very High	0.81
APEX	Low	1.10
PLEX	Low	1.09
LTEX	Nominal	1.00
TOOL	Very High	0.78
SITE	High	0.93
SCED	High	1.00
Total (multiplication):		1.27256

2.2.3 Final results

Basing ourselves on the values calculated above and on the equations shown at the beginning of the chapter, these are our final results:

$E = B + 0.01 \times \sum_{j=1}^{5} SF_j = 0.91 + 0.01 \times 18.96 = 1.0996$

$EAF = \prod_{i=1}^{17} CD_i = 1.27256$

$Effort_{\text{avg}} = A \times EAF \times KSLOC^{E} \ = 2.94 \times 1.27256 \times (9.338)^{1.0996} = 43.64PM = 44PM$

$Effort_{\text{high}} = A \times EAF \times KSLOC^{E} \ = 2.94 \times 1.27256 \times (13.601)^{1.0996} = 65.99PM = 66PM$

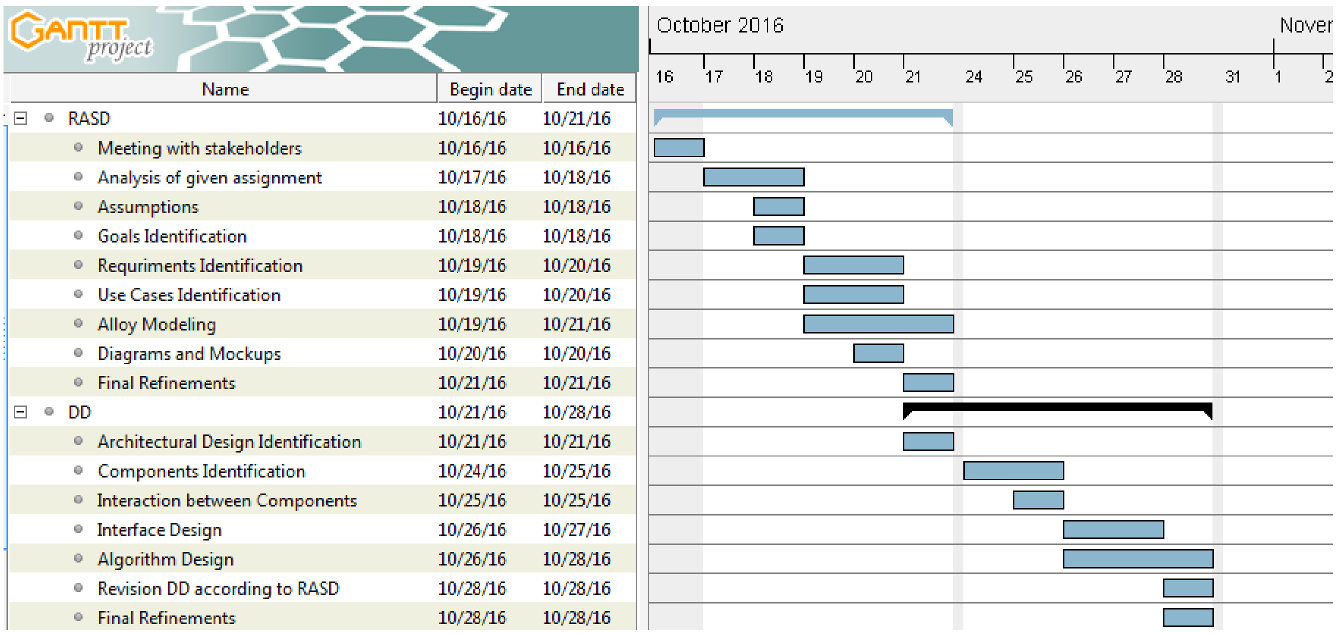
$F = D + 0.2 \times (E - B) = 0.28 + 0.2 \times (1.0996 - 0.91) = 0.31792$

$Duration_{\text{avg}} = C \times Effort^{F} = 3.67 \times 44PM^{0.31792} = 12.22months$

$Duration_{\text{high}} = C \times Effort^{F} = 3.67 \times 66PM^{0.31792} = 13.90months$

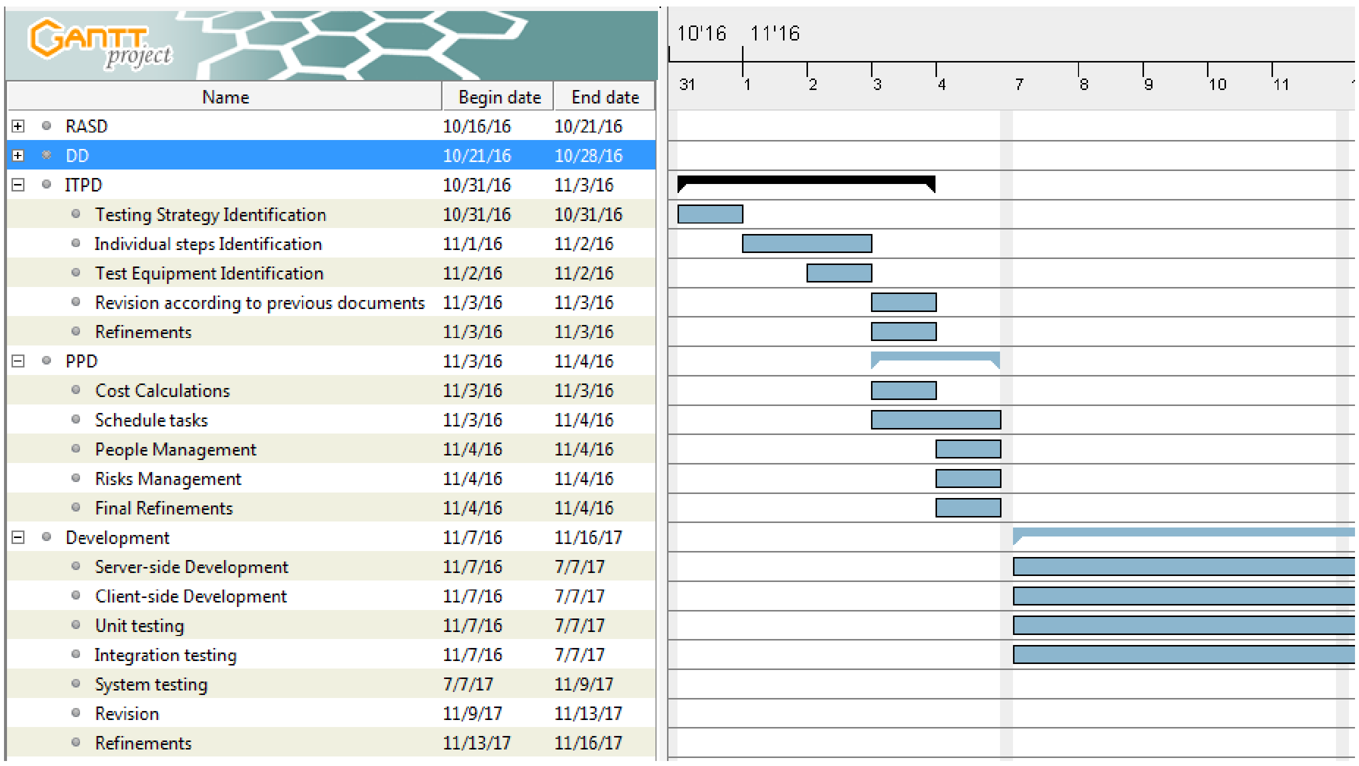
3 Schedule

In this section the detailed schedule is presented as Gantt diagram.
It's important to mention that the schedule was made taking into account that the project will be done by professional full-time workers. This said that the documentation part of the project should be done in fewer days than it actually was. However number of hours for each documentation part is the average number of hours our team actually spent.
Some diagrams are cut in order to mantain readability.



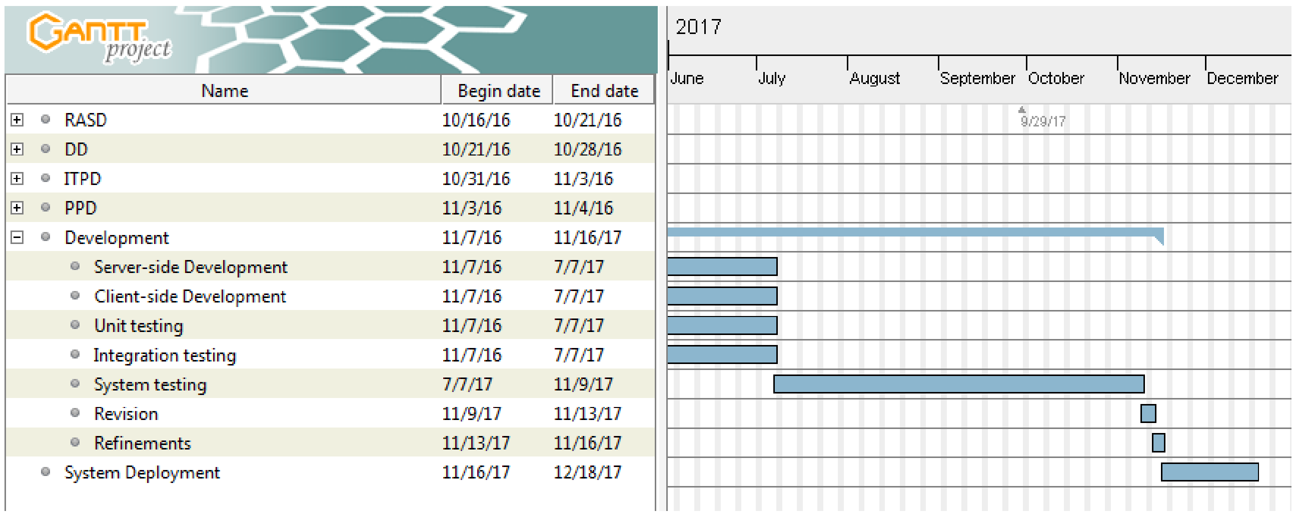
Name	Begin date	End date
RASD	10/16/16	10/21/16
Meeting with stakeholders	10/16/16	10/16/16
Analysis of given assignment	10/17/16	10/18/16
Assumptions	10/18/16	10/18/16
Goals Identification	10/18/16	10/18/16
Requriments Identification	10/19/16	10/20/16
Use Cases Identification	10/19/16	10/20/16
Alloy Modeling	10/19/16	10/21/16
Diagrams and Mockups	10/20/16	10/20/16
Final Refinements	10/21/16	10/21/16
DD	10/21/16	10/28/16
Architectural Design Identification	10/21/16	10/21/16
Components Identification	10/24/16	10/25/16
Interaction between Components	10/25/16	10/25/16
Interface Design	10/26/16	10/27/16
Algorithm Design	10/26/16	10/28/16
Revision DD according to RASD	10/28/16	10/28/16
Final Refinements	10/28/16	10/28/16

Figure 1: Schedule for RASD and DD parts



Name	Begin date	End date
RASD	10/16/16	10/21/16
DD	10/21/16	10/28/16
ITPD	10/31/16	11/3/16
Testing Strategy Identification	10/31/16	10/31/16
Individual steps Identification	11/1/16	11/2/16
Test Equipment Identification	11/2/16	11/2/16
Revision according to previous documents	11/3/16	11/3/16
Refinements	11/3/16	11/3/16
PPD	11/3/16	11/4/16
Cost Calculations	11/3/16	11/3/16
Schedule tasks	11/3/16	11/4/16
People Management	11/4/16	11/4/16
Risks Management	11/4/16	11/4/16
Final Refinements	11/4/16	11/4/16
Development	11/7/16	11/16/17
Server-side Development	11/7/16	7/7/17
Client-side Development	11/7/16	7/7/17
Unit testing	11/7/16	7/7/17
Integration testing	11/7/16	7/7/17
System testing	7/7/17	11/9/17
Revision	11/9/17	11/13/17
Refinements	11/13/17	11/16/17

Figure 2: Schedule for ITPD and PP parts

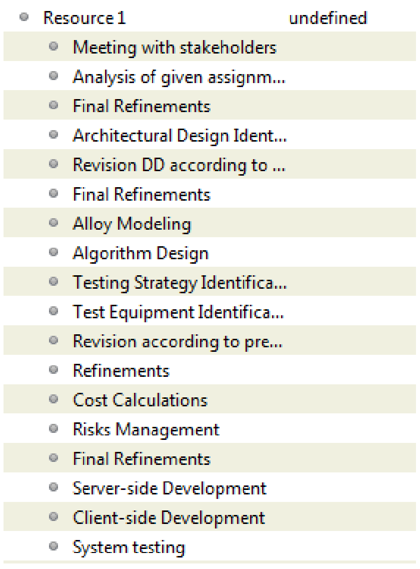


Name	Begin date	End date
RASD	10/16/16	10/21/16
DD	10/21/16	10/28/16
ITPD	10/31/16	11/3/16
PPD	11/3/16	11/4/16
Development	11/7/16	11/16/17
Server-side Development	11/7/16	7/7/17
Client-side Development	11/7/16	7/7/17
Unit testing	11/7/16	7/7/17
Integration testing	11/7/16	7/7/17
System testing	7/7/17	11/9/17
Revision	11/9/17	11/13/17
Refinements	11/13/17	11/16/17
System Deployment	11/16/17	12/18/17

Figure 3: Schedule for Development and System Deployment parts

4 Resource allocation

This section presents how the tasks defined in the Schedule section will be devided between three team members.



Resource 1	undefined
Meeting with stakeholders	
Analysis of given assignm...	
Final Refinements	
Architectural Design Ident...	
Revision DD according to ...	
Final Refinements	
Alloy Modeling	
Algorithm Design	
Testing Strategy Identifica...	
Test Equipment Identifica...	
Revision according to pre...	
Refinements	
Cost Calculations	
Risks Management	
Final Refinements	
Server-side Development	
Client-side Development	
System testing	

Figure 4: Tasks for Resource1

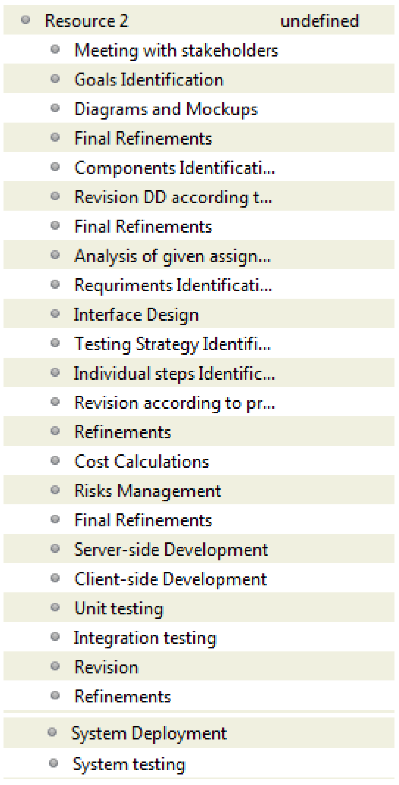

Resource 2 undefined
- Meeting with stakeholders
- Goals Identification
- Diagrams and Mockups
- Final Refinements
- Components Identificati...
- Revision DD according t...
- Final Refinements
- Analysis of given assign...
- Requriments Identificati...
- Interface Design
- Testing Strategy Identifi...
- Individual steps Identific...
- Revision according to pr...
- Refinements
- Cost Calculations
- Risks Management
- Final Refinements
- Server-side Development
- Client-side Development
- Unit testing
- Integration testing
- Revision
- Refinements
- System Deployment
- System testing

Figure 5: Tasks for Resource2

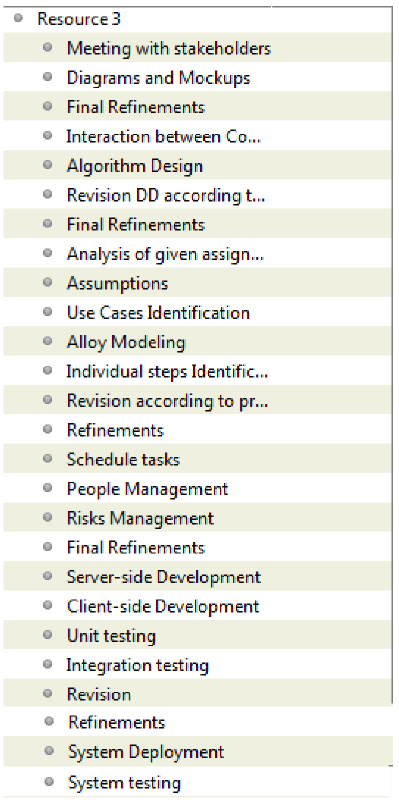

Resource 3
- Meeting with stakeholders
- Diagrams and Mockups
- Final Refinements
- Interaction between Co...
- Algorithm Design
- Revision DD according t...
- Final Refinements
- Analysis of given assign...
- Assumptions
- Use Cases Identification
- Alloy Modeling
- Individual steps Identific...
- Revision according to pr...
- Refinements
- Schedule tasks
- People Management
- Risks Management
- Final Refinements
- Server-side Development
- Client-side Development
- Unit testing
- Integration testing
- Revision
- Refinements
- System Deployment
- System testing

Figure 6: Tasks for Resource3

5 Risk management

Being able to manage and properly recover from any issue that may arise working on a project is a particularly important activity to consider. In order to be prepared, a proactive strategy is chosen to be applied. In particular, what follows is a survey of the identified risks, with details about their probability to happen (Low, Moderate, High) and the magnitude of their impact (Negligible, Marginal, Serious, Catastrophic). With these parameters it is possible to focus on the most critical issues and the contingency that has to be devised for them.

Personnel shortfall

Description It may happen that one of the team members becomes unavailable for a certain amount of time, mainly due to:

- temporary illness;
- resignation presented to the company.

Probability Moderate: it is not a frequent situation, but at least once in the project lifespan an illness situation is likely to happen.

Impact Serious: the project schedule is likely to be adapted and if it happens close to a critical phase an improper handling may cause important damages to the project itself.

Contingency plan In general, the team should be organized such that every task is known enough by more than one person, so that even in case of an absence, people can be reassigned to cover all the tasks. However, since our team is very restricted, we know it may become more difficult to address this requirement, and some more extra time should be allocated to the project in order to recover from this situations. Another important precaution is that the group should be kept up to date with any difficulty and delays each member encounters, so that the work could be rescheduled as soon as possible.

Bad external frameworks/libraries

Description For the development of the system, many frameworks and libraries will be taken or bought from online repositories or other providers. Doing this, we rely on those components to properly work. It may therefore happen that some unexpected fault related to them happens, at any point of the development.

Probability Moderate: almost every external component, especially if open-source, has major and minor issues.

Impact Serious: the development has to stop or slow down and the bug must be fixed, by its provider or directly by us. If this is impossible or takes too much time, the component must be substituted and/or the functionalities it provided must be re-developed. Note that a substitution with another bought component implies its integration and an eventual adjustment of the software components connected to it.

Contingency plan The external components that will be used for the project must be properly checked as soon as possible for every needed functionality. This of course is not enough to solve the problem, since testing can never be complete. For this reason, two strategies are combined: on one hand, when an external component is chosen, a list of some possible alternatives should be draft, evaluating them on the functionalities provided and on the smoothness with which they can substitute the chosen one. On the other hand, an extensive check on the open and reported issues for the chosen component must be performed before its insertion, in order to discover and analyse potential issues in advance. In addition, we must always take into account if an assistance plan is offered by the providers.
In a macro-analysis, the following components have been identified as critical:

- **Localization service.** We rely on an external localization service for critical operations, such as the tracking of the company's car, and a fault can cause large economic damages. However, this services are well spread and widely used by many applications and offer a very high reliability, so we don't expect big issues to get raised.
- **Payment service.** The payment service results critical too since it handles the customers' money. However, as for the localization service, it is a well spread and used service, thus no big issues are expected.
- **Car sensors system.** Being the core element of our system, the cars need to have the lowest error probability possible. For this reason, a particular attention must be paid to the car sensors system. In addition, we must consider that this is not a wide-spread system, so on one hand a particularly accurate testing must be performed during the whole development, while on the other hand an assistance plan with the provider company will be subscribed.

Changes in external services interfaces

Description Modifications may occur in the ways we connect to some external service provider. In particular, this modifications may not respect retrocompatibility and force us to modify our system too.

Probability Low: non-retrocompatible modifications are not recommended and respectable providers do not apply them.

Impact Marginal: we may need to modify the more external components of our architecture, but with a proper design the changes should be contained.

Contingency plan The contingency time reserved in the scheduling phase for exceptional situations should consider also this risk. In addition, whenever possible it is better to sign specific contracts with the service providers that guarantees compatibility and support over time.

Data loss

Description Some loss of data or code might happen due to the failure of one or more machines.

Probability Low: modern hardware has a very low failure ratio.

Impact Catastrophic: a loss of important data about the employment of the cars can lead to important economic consequences.

Contingency plan This is a very standard problem to tackle. Data are preserved using a backup system, often included in the cloud services solution. The backup interval will be scaled on the average frequency of changes to the data itself: for instance, a daily backup is considered to be sufficient for static users' and cars' data, while a hourly backup is more suited for the actual employment data.
The software code will be preserved instead adopting cloud repositories as storage and versioning systems. Occasional backups of the developers machines may be programmed.

Changes in laws or regulations

Description National traffic laws and car sharing regulations may change in the future. Besides, the procedures used to validate the licenses may be subjected to modifications.

Probability Low: an analysis of the past variations to this subjects reveals that only few and rare modifications have been made in the past years, and almost all of them do not affect our application domain.

Impact Serious/Catastrophic: highly dependent on the nature of the changes, it may require from few modifications to a general reorganization of our system. It may also require us to withdraw our system from the market.

Contingency plan The only precaution we can take is to constantly keep ourselves updated on the matter and analyze the eventual modifications as soon as they are proposed. In addition, a good design and development will help to perform quick and low-cost modifications if needed.

Wrong functionalities or user interface

Description After the development, the functionalities or the user interface for accessing them do not fulfill the needs of the stakeholders.

Probability Low: since an explicit validation is required for the feasibility study and the RASD, there is less probability for this to happen.

Impact Serious: some parts of the system and its UI might be restructured or completely redesigned, causing considerable delays in the delivery of the system.

Contingency plan A validation is requested not only for the RASD, but also for the DD, organizing if needed a meeting to explain our main stakeholders the key points and especially the limits of the proposed architecture. After this phase, periodical meeting are scheduled, to get internal validations of the components already implemented. This meetings will be more frequent as the project progresses, in order to have feedbacks on working pieces of the system.
The UI will also be validated, firstly with the mockups presented in the RASD, and later in the same meetings described before.
Alpha and beta testing will be planned as well, to tackle also possible usability issues.

Even though the previous analysis, we are aware that it is impossible to consider every unpredictable risk in advance. For these risks, a contingency time is allocated during the scheduling and resource allocation phase of the project planning.

In conclusion, we encourage every stakeholder and potential user to point out any risk or issue as soon as it is detected, providing them any useful information to be contacted.

6 Effort spent

Abbud, Patricia around 11 hours of work;

Andreoli Andreoni, Maddalena around 16 hours of work;

Cudrano, Paolo around 14 hours of work.

7 Appendix

List of Figures