Politecnico di Milano

Software engineering 2

PowerEnjoy
Project Plan v1.0

Alessandro Caprarelli	874206
Roberta Iero	873513
Giorgio De Luca	875598

January 19, 2017

Contents

Chapter 1

Introduction

1.1 Revision history

Table 1.1: Revision history

Version	Date
v1.0	22/01/2017

1.2 Purpose

This document represents the Project Plan for the development of the application of Power EnJoy. The aim of this document is to define the approach and the schedule of the project, in order to give a guideline to follow to the project team. The decisions taken in this document take in account the requests from both the stakeholders and the development team.

1.3 Scope

The aim of the project PowerEnjoy is to provide an automated service of car sharing. After a registration, a client can hire a car near him/her through the web or mobile application and he/she can enjoy of all the extra services offered. The exact position of the client is determined by the GPS signal of the client's device or it's allowed to manually insert a specific address. So the system displays all the available cars

in the client's close area. Then the client could make a reservation of a car, after which the system notifies the client with a message of confirmation with the car identifier. If the reservation procedure successfully ends the chosen car won't be available anymore for other clients. Moreover a client cannot hire more than one car at the same time. After the reservation the client has at most one hour to reach the car, when this time expires the system gives a penalty to the client and the car, previously hired, is available again for other clients. The system allows the client to cancel his/her reservation. When the client reaches the car, he/she can tell the system that is nearby through a specific button in the application and he/she starts to pay as soon as the engine ignites. During the travel the system supervises the current charge of the car and notifies it to the client through a screen located in the car. The system stops charging the amount of money that the client has to pay when he/she communicates through the application his/her decision to stop the rent. When the car is parked in a safe area and the client exits, the system locks the car automatically and starts the procedure of payment. The client is notified with the result of this procedure through an SMS, including the final fare.

1.4 Definition, acronyms, abbrevations

1.4.1 Definition

- **Guest client:** a person that is not already registered in the system or that has to log in.
- **Registered client:** a person who has valid access credentials to log in the system.
- **System administrator:** privileged user, in charge of managing administration processes and of updating business logic.
- **Reservation:** it is the action performed by a registered client that allow him/her to reserve an available car for maximum one hour.
- **Journey time = travel time:** time elapsed since the user starts the engine to the user parks the car and terminates the journey.
- **Available car:** a car that is not reserved by any user and has enough charge to be rented.
- **Unavailable car:** a car that is already reserved or damaged, so impossible to reserve.

- **Gps navigation:** it is the navigation system that is included in the car on board system. It could be used by the user to find direction to the final destination.
- **Final destination:** address where the user wants to go.
- **Safe area:** the region where is permitted to park and leave a car once the rent is terminated.
- **Power grid station:** the area where it's allowed users to park the cars, leaving them attached to the power grid.

1.4.2 Acronyms

- **ITP:** Integration Test Plan
- **DD:** Design Document
- **RASD:** Requirements Analysis and Specification Document
- **API:** Application Programming Interface
- **UI:** User Interface
- **DBMS:** Data Base management system
- **COCOMO:** COnstructive COst MOdel
- **SF:** Scale Factor
- **UFP:** Unadjusted Function Point

1.4.3 Abbrevetations

1.5 Reference documents

- RASD v1.1
- DD v1.0
- ITPD v1.0
- PowerEnjoy specification document (assignment).

- IEEE Std 1016tm-2009 Standard for Information Technology - System Design - Software Design Descriptions.

1.6 Document overview

The ITP is composed of five sections:

- Introduction: this section defines the goal of the document and the main characteristics of the project of PowerEnJoy.
- Cost estimation: in this second part there is the cost estimation of the project, done with two different methods: Function Points and COCOMO II. In order to give a greater understanding of the subject, there is also a brief introduction to both the methods used.
- Project Tasks and Schedule: in this section i's provided a schedule of the different phases of the project.
- Resources and Tasks allocation: the fourth section contains the allocation of the resources and the tasks among the components of the team.
- Project Risks: this last section describes the possible risks the project may be exposed to, distinguished by type.

Chapter 2

Cost estimation

2.1 Function point

2.1.1 Brief introduction

The Function Point estimation approach is based on the principle of extracting functions from a software, to classify them using a well defined set of classes and estimating their complexity.

This kind of estimation is extremely useful since it can be done at a very early stage of a project life-cycle, ideally after the implementation of the RASD.

This estimation is a single number called UFP that can be computed using a simple formula.

A high-level procedure that explains how to calculate this number is the following:

1. Classify each function of the software to one of these five possible classes called Function Types (explained in detail later):
    - Internal logic files
    - External logic interfaces files
    - External Inputs
    - External outputs
    - External inquiries
2. For each function a complexity is defined, and it can be:

- Low
- Average
- High

The choice of the complexity is based on the analysis of the quantity of data processed by each function and takes into account also the type of interaction required between different components.

3. Calculate the UFP using the formula:

$$\sum_{f \in F, c \in C} ((\sharp \text{ of function of type } f \text{ and complexity } c) * (\sharp \text{ weight for type } f \text{ and complexity } c))$$

where F=ILF, ELF, EI, EO, EIQ and C=Low, Average, High.
Refer to this table to determine the proper weight for each type and complexity:

Table 2.1: UFP Complexity Weights

Function type	Low	Average	High
Internal Logic files	7	10	15
External Interfaces files	5	7	10
External Inputs	3	4	6
External Ouputs	4	5	7
External Inquiries	3	4	6

Further manipulation of the UFP can be done in order to use it in Cost Estimation Models such as COCOMO, but this will be explained later.

2.1.2 Internal Logic Files

The application manages a database which is used to store different kind of entities, each one of them has its own data structure.

These entities are: registered Client, system administrator, car, safe area, reservation/request, rent, transaction.

Table 2.2: ILF recap

ILF	**Complexity**	**FP**
Registered client	Low	7
System administrator	Low	7
Car	Low	7
Safe area	Average	10
Reservation/Request	High	15
Rent	Average	10
Transaction	Average	10
Total:		**66**

2.1.3 External interface files

The application of Power EnJoy must interact with some external services in order to fulfill its main goals.

The external services are about: driving License, maps, payment gateway, SMS gateway, push notification gateway, customer support.

Table 2.3: EIF recap

ELF	**Complexity**	**FP**
Driving License	Low	5
Maps	High	10
Payment gateway	Average	7
SMS gateway	Low	5
Push notification gateay	Low	5
Customer support	Low	5
Total:		**37**

2.1.4 External Inputs

The application has to handle external interactions due to the Registered Client's actions or to the System Administrator's actions.

These external interactions are: registration, login, logout, modify profile, make a reservation, choose a destination, terminate a rent, manage cars (System Administrator).

Table 2.4: EI recap

EI	**Complexity**	**FP**
Registration	Low	3
Login	Low	3
Logout	Low	3
Modify profile	Low	3
Make a reservation	High	6
Chose a destination	Average	4
Terminate a rent	High	6
Cancel a reservation	High	6
Manage cars (SystemAdmin)	Average	4
Total:		**38**

2.1.5 External Outputs

The application generates the following outputs, result of an internal computation on some data, to the Registered Client or to the System Administrator:

Table 2.5: EO recap

EO	**Complexity**	**FP**
Best destination for getting a discount	High	7
List of available cars	Low	4
Notifications	Low	4
Total:		**38**

2.1.6 External Inquiries

The application generates the following outputs to the Registered Client or to the System Administrator:

Table 2.6: EInq recap

EInq	**Complexity**	**FP**
Show Registered Client profile	Low	3
Show System Administrator profile	Low	3
Show history of past rents	Low	3
Show current cost, charge and maps through the OnBoard application	Average	4
Total:		**14**

2.1.7 Recap

The following table summarizes the amount of FP for each type and the overall sum that corresponds to the final UFP value.

Table 2.7: Function point recap

Function type	**Value**
Internal logic files	66
External interfaces files	37
External inputs	38
External outputs	15
External inquiries	13
Total:	**169**

$$\sum_{f \in F, c \in C} ((\sharp \text{ of function of type } f \text{ and complexity } c) * (\sharp \text{ weight for type } f \text{ and complexity } c)) = 169$$

2.2 COCOMO

2.2.1 Brief introduction

The aim of using COCOMO is to obtain an estimation of the effort needed to develop the software in analysis. The effort is defined by the following formula:

$$\text{Effort} = A \cdot SLOC^{E} \cdot \prod_{i=1}^{n} EM_i$$

The elements considered to define the values for the estimation are based not only on technical drivers like SLOC or complexity of the software, but also on intrinsic variables like workforce and business processes.

2.2.2 Lines of Codes

Code size is expressed in thousands of source lines of code (KSLOC). In order to estimate this parameter we consider the value of the UFP and we convert it using the standard value indicated in COCOMO II for JavaScript. So we obtain the following values:

- Lower bound: $SLOC = 169 \cdot 31 = 5239 = 5,239 KSLOC$
- Average bound: $SLOC = 169 \cdot 31 = 7943 = 7,943 KSLOC$
- Upper bound: $SLOC = 169 \cdot 63 = 10647 = 10,647 KSLOC$

From now on we consider for our analysis the average value of 7,943 KSLOC.

2.2.3 Scale Drivers

Scale drivers are factors used to calculate the exponent E of the Effort equation defined above. E can be computed using this equation:

$$E = B + 0.01 \cdot \sum_{j=1}^{f} SF_j$$

We can distinguish three different cases:

- If $E < 1.0$, the project exhibits economies of scale.

- If $E = 1.0$, the economies and diseconomies of scale are in balance.
- If $E > 1.0$, the project exhibits diseconomies of scale.

In this section five factors are considered to evaluate E:

- Precedentedness: it measures the experience related to the kind of project and development, that derives from similarity of projects previously developed. We set this parameter to low.
- Development flexibility: it expresses the capacity of development team to adapt itself to changes during the development process. In particular it measures how much the software needs to be accurate with the respect to the initial requirements and to the external interfaces. We set this scale driver to nominal.
- Risk resolution: it expresses the amount of effort and budget to be invested in order to predict and manage risks that can occur during the life cycle of the software. We set this parameter to nominal.
- Team cohesion: it represents the capabilities of all the actors involved in the development process to communicate and interact with the aim of reach the result expected. We set this scale driver to very high.
- Process maturity: it's a parameter indicating the ability to manage development processes, like defect management, quality assurance, change management, etc. In particular it gives a measure of how much sophisticated the development of the project is, basing the analysis on some general criteria. We set this scale driver to high.

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC:	Thoroughly unprece-dented	Largely unprece-dented	Somewhat unprece-dented	Generally familiar	Largely familiar	Thoroughly familiar
SF_i:	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_i:	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_i:	7.07	5.65	4.24	2.83	1.41	0.00
TEAM:	Very difficult interactions	Some difficult in-teractions	Basically cooperative interac-tions	Largely cooperative	Highly cooperative	Seamless interactions
SF_i:	5.48	4.38	3.29	2.19	1.10	0.00
PMAT:	The estimated Equivalent Process Maturity Level (EPML) or					
	SW-CMM Level 1 Lower	SW-CMM Level 1 Upper	SW-CMM Level 2	SW-CMM Level 3	SW-CMM Level 4	SW-CMM Level 5
SF_i:	7.80	6.24	4.68	3.12	1.56	0.00

Figure 2.1: Scale factor values, SF_i, for COCOMO II Models

In the following table we recap the values we have chosen for the Scale Factors.

Table 2.8: Scale factors recap

Scale Driver	Selected Factor	Value
Precedentedness	Low	4.96
Development flexibility	Nominal	3.04
Risk resolution	Nominal	4.24
Team cohesion	Very High	3.12
Process maturity	High	3.04
Total:		**17.55**

2.2.4 Cost Drivers

Cost Drivers are model factors which aim is to take into account the main characteristics of the software development, characteristics that influence the cost. The

main Cost Drivers and their standard respective multipliers that we consider are:

- **Required Software Reliability:** this measures the extent to which the software must perform its functions over a period of time. We set this parameter to nominal.

RELY Descriptors:	Slight inconvenience	Low, easily recoverable losses	Moderate, easily 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

Figure 2.2: RELY cost driver

- **Database size:** this measure is related to the dimension of the database. The reason to do it is to consider the effort required to generate the test data that will be used to exercise the program. In our case its value is nominal.

DATA Descriptors:		Testing DB bytes/Pgm SLOC < 10	$10 \leq$ D/P ≤ 100	$100 \leq$ D/P $\leq$ 1000	D/P ≥ 100	
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

Figure 2.3: DATA cost driver

- **Product complexity:** the complexity is derived consider five different areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations. We set this parameter to high.

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

Figure 2.4: CPLX cost driver

- **Required reusability:** this cost driver estimates the effort needed to create new components that have to reuse the current project. We set set this parameter to high.

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

Figure 2.5: RUSE cost driver

- **Documentation match to life-cycle needs:** it measures the suitability of the documentation of the project with its real life usage characteristics. We set this parameter to nominal.

DOCU Descriptors:	Many life-cycle needs unconvered	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

Figure 2.6: DOCU cost driver

- **Execution time constraint:** this measure states an execution time constraint and is expressed in terms of the percentage of available execution time expected to be used by the system or by a subsystem. We set this parameter to nominal.

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

Figure 2.7: TIME cost driver

- **Storage constraint:** this parameter describes the expected amount of storage usage with respect to the availability of the hardware. We set this parameter 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 Level	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	n/a	1.00	1.05	1.17	1.46

Figure 2.8: STOR cost driver

- **Platform Volatility:** with the term platform we indicates both software and hardware that the software has to call in order to pursue its tasks. This factor measures the probability of our platform to be changed. We set this parameter to high.

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 Level	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	0.87	1.00	1.15	1.30	n/a

Figure 2.9: PVOL cost driver

- **Analyst Capability:** this measures analysts' abilities, capacity of communication and efficiency. We set this parameter to very high.

ACAP Descriptors:	15^{th} percentile	35^{th} percentile	55^{th} percentile	75^{th} percentile	90^{th} 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

Figure 2.10: ACAP cost driver

- **Programmer Capability:** this factor measures the capability of programmers to work as individuals and in a team. We set this parameter to very high.

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

Figure 2.11: PCAP cost driver

- **Application Experience:** this cost driver depends on the level of experience that the project team have to develop the software system or subsystem. We set this parameter to nominal.

APEX Descriptors:	≤ 2 months	6 months	1 year	3 year	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

Figure 2.12: APEX cost driver

- **Platform Experience:** this factor measures the experience on the usage of the platform. We set this parameter to nominal.

PLEX Descriptors:	≤ 2 months	6 months	1 year	3 year	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

Figure 2.13: PLEX cost driver

- **Language and Tool Experience:** this is a measure of the level of programming language and software tool experience of the project team developing the software system or subsystem. We consider this value as nominal.

LTEX Descriptors:	≤ 2 months	6 months	1 year	3 year	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

Figure 2.14: LTEX cost driver

- **Personnel continuity:** it expresses a measure of the project's annual personnel turnover. We consider this value as very low.

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

Figure 2.15: PCON cost driver

- **Usage of Software Tools:** this parameter is set considering the effort needed to develop and integrate all the tools that the software requires. The value of this parameter is 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 Level	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.17	1.09	1.00	0.90	0.78	n/a

Figure 2.16: TOOL cost driver

- **Multi-site development:** this cost driver has been chosen considering two factors: site collocation and communication support. We set this parameter to high.

SITE: Collocation Descriptors: **SITE: Communications Descriptors:**	International Some phone, mail	Multi-city and Multi-company Individual phone, FAX	Multi-city or Multi-company Narrow band email	Same city or metro. area Wideband electronic communication	Same building or complex Wideband elect. comm. occasional video conf.	Fully collocated 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

Figure 2.17: SITE cost driver

- **Required development schedule:** this cost driver measures the schedule constraints imposed on the project team developing the software. The ratings are defined in percentage of schedule stretch-outs or accelerations with respect to a nominal schedule for a project requiring a given amount of effort. We set this parameter to nominal.

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

Figure 2.18: SCED cost driver

2.2.5 Cost driver recap

In the following table we recap the values we have chosen for the Cost Drivers.

Table 2.9: Cost factors recap

Cost Driver	**Selected Factor**	**Value**
Required Software Reliability	Nominal	1.00
Database Size	Nominal	1.00
Product Complexity	High	1.17
Required Reusability	High	1.07
Documentation match to life-cycle needs	Nominal	1.00
Execution Time Constraint	Nominal	1.00
Main Storage Constraint	Nominal	1.00
Platform Volatility	High	1.15
Analyst Capability	High	0.85
Programmer Capability	High	0.88
Application Experience	Nominal	1.00
Platform Experience	Nominal	1.00
Language and Tool Experience	Nominal	1.00
Personnel Continuity	Very Low	1.29
Usage of Software Tools	High	0.90
Multi-site Development	High	0.93
Required Development Schedule	Nominal	1.00
Product:		**1.16**

$$\prod_{i=1}^{n} EM_i = 1.16$$

2.2.6 Effort computation

Using the values obtained in the previous section we can estimate duration and effort. Given the standard value of COCOMO II A = 2.94, B = 0.91, C=3.67 and D= 0.28, we have:

$$E = B + 0.01 \cdot \sum_{j=1}^{5} SF_j = 0.91 + 0.01 \cdot 17.55 = 1.0855$$

$$EAF = \prod_{i=1}^{n} EM_i = 1.16$$

$$\text{Effort} = A \cdot EAF \cdot KSOLC^E = 2.94 \cdot 1.16 \cdot 7,943^{1.0855} = 32.33 \text{ person/month}$$

$$(D + 0.2 \cdot (E - B)) = (0.28 + 0.2 \cdot (1.0855 - 0.91)) = 0.3151$$

$$\text{Duration} = [C \cdot (PM)^{D+0.2\cdot(E-B)}] = [3.67 \cdot (32.33)^{0.3151}] = 10.97 = 11 \text{ months}$$

$$N = \text{ number of people needed } = \text{ Effort/Duration } = 32.33/11 = 2.93 = 3 \text{ people}$$

Chapter 3

Project Tasks and Schedule

In this section we define the schedule of the project. In order to do that we provide a Gantt diagram to show the tasks and their respective deadlines.

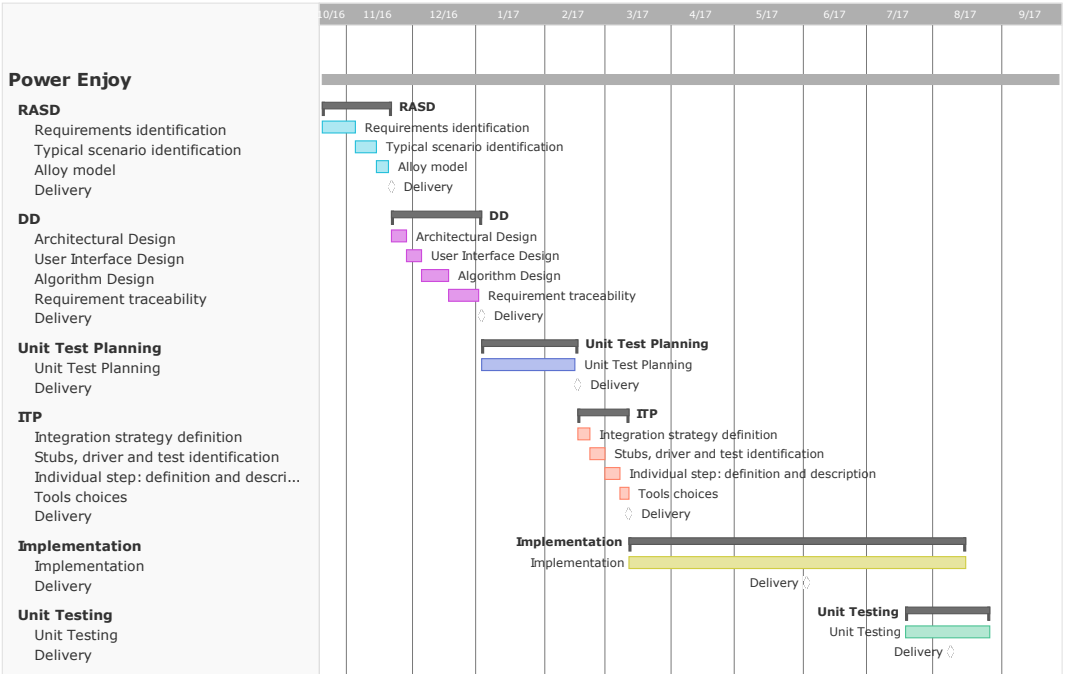

Figure 3.1: Gantt

Chapter 4

Resources and Tasks Allocation

The following diagram shows the allocation of resources and tasks between the three components of the development team. The work is scheduled more or less fairly to the three components. The reason of doing so is that especially the first phases of the project, during which is necessary to interact and to take decisions together, is difficult to assign sub tasks. Concerning the implementation and the testing part, the work is done in parallel by the members of the team, even though every person has his/her own sub-tasks.

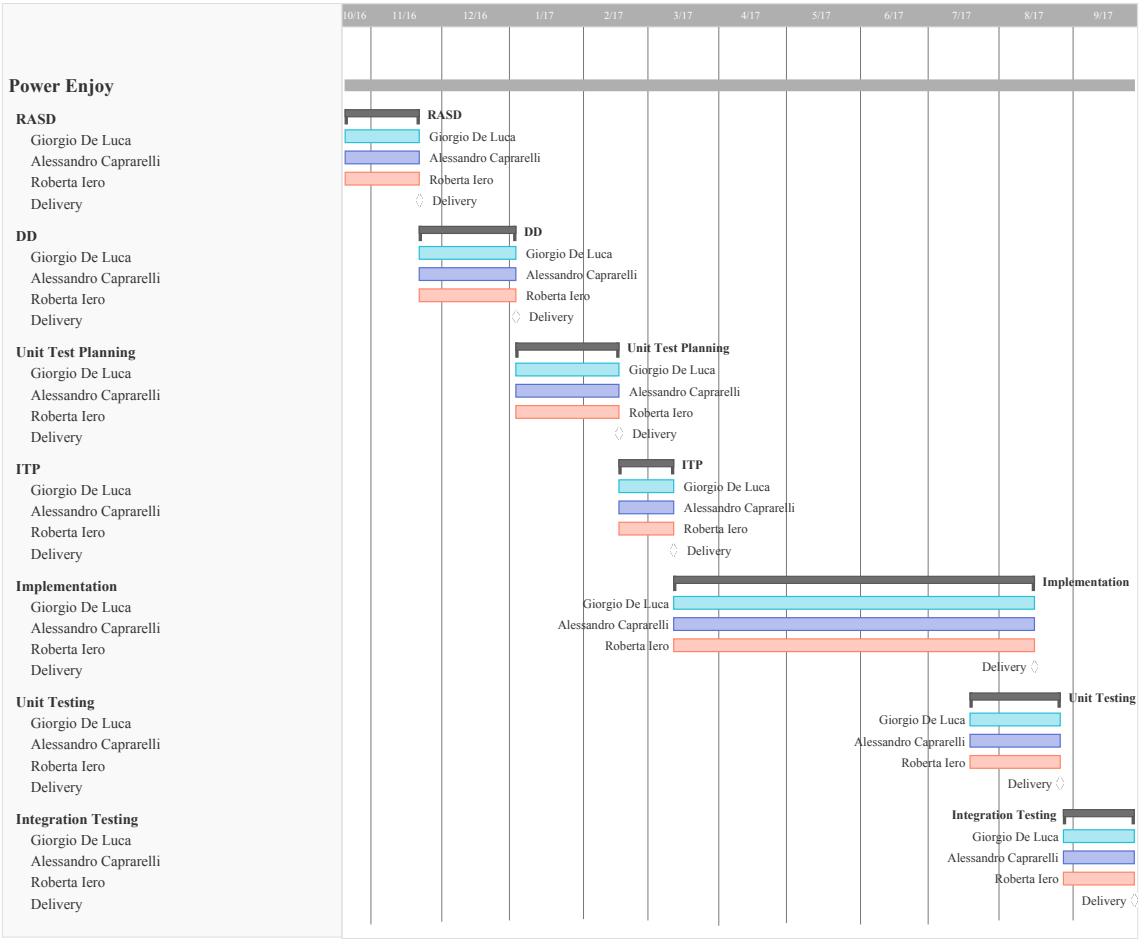

Figure 4.1: Resources allocation

Chapter 5

Project risks

In order to classify the possible risks occurring during the development process, the following probabilities related to the occur of the events are defined:

- Low
- Moderate
- High

The impacts of unexpected events are classified on the basis of the following scale:

- Negligible
- Marginal
- Serious
- Critical
- Catastrophic

Table 5.1: Project risks

Risk	Probability	Impact
Requirements volatility	Moderate	Serious
Team cohesion	Low	Critical
Unusability of external components	Moderate	Critical
Security issues	Low	Critical
Downtime	Low	Serious
Delays on project deadlines	Moderate	Serious
Lack of attractiveness of the application	Moderate	Catastrophic
Integration Test Failure	Low	Critical
Bankruptcy	Low	Catastrophic
Competitors	Moderate	Critical
Use of other transports	Moderate	Critical
Changes of administrative rules and policies	Low	Critical
Increase of taxes and costs of maintenance	Moderate	Marginal

Considering the analysis done in the previous table, we illustrate for each kind of expected risk a possible strategy to adopt in order to react, whenever it's possible.

Risk	Strategy
Requirements volatility	Integrate the missing information and persuade the stakeholders to change deadlines of releases
Team cohesion	A possible strategy is to clearly assign the task to each member of the team and try to encourage the communication, to maintain a relaxed environment.
Unusability of external components	Solve connection problems or, in case of external services, notify as soon as possible the problem to whom it may concern
Security issues	Generate a lot of focused test to analyze the behaviour of the system in different scenarios that represents bad possible behaviours of the clients

Downtime	Understand the problem the causes the downtime, fix it and make the system available again
Delays on deadlines of the project	Explain the causes of the delays to the stakeholders, support them in their decisions and try to not going beyond in the future deadlines
Lack of attractiveness of the application	Do a good analysis of the products of the competitors in order to identify their weaknesses and to add more features to capture the clients' attention
Integration Test Failure	Execute several tests on each single component of the system
Bankruptcy	A possible strategy to prevent it is to do a correct cost estimation and a good allocation of the tasks and the resources
Competitors	Add more features to the project to let it become more attractive and at the same time do a good advertising campaign
Use of other kind of means of transport	Impossible to prevent. Possible solutions: acquire the new incoming business model; sell the society in time or convert it for another business model
Changes of administrative rules and policies	Lobby activities
Increase of taxes and costs of maintenance	Take into account during the cost estimation the possibility to save a part of the budget in order to face this problem

Chapter 6

Used tools and working hours

6.1 Used tools

The tools used to create this document are the following:

- MikTex 2.9: to format the document using LaTex.
- teamgantt.com: to create Gantt and resources allocation diagrams.

6.2 Working hours

	Alessandro	Roberta	Giorgio
14/1	2	2	2
15/1	1	1	1
16/1			
17/1	3	3	3
18/1	1	1	1
19/1	3	3	3
20/1	3	3	3
21/1	2	2	2
Total	15	15	15