PowerEnJoy
Software Engineering II

Project Plan Document

Giovanni Scotti, Marco Trabucchi

Document version: 1
January 17, 2017

Contents

Section 1

Introduction

1.1 Purpose and Scope

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

In the first section of the document, two complementary estimation models, *Function Point Analysis (FPA)* and *COCOMO II*, are going to be used in order to support the estimation process; the result will be an estimate of the number of lines of source code (SLOC) to be written and the average effort required for the development process itself.

In the second section, the organizational structure of the project plan will be laid down; here will be defined a possible schedule to cover all activities within the deadline and with a proper redistribution of tasks among the team members.

The last section of the document contains an analysis of all kinds of risks that the project could be facing throughout its life-cycle, from the development phase to the deployment phase, up to the maintenance and support phase. Here are also defined the safety measures, both reactive and preemptive, to face the eventual occurrence of the risks mentioned above.

1.2 Definitions, Acronyms and Abbreviations

COCOMO: COnstructive COst MOdel

DD: Design Document

EI: External Inputs

EIF: External Interface Files

EO: External Outputs

EQ: External Inquiries

FP: Function Point

FPA: Function Point Analysis

ILF: Internal Logic Files

ITPD: Integration Test Plan Document

LDF: Language-Dependent Factor

RASD: Requirements Analysis and Specification Document

SLOC/KSLOC: Source Lines Of Code / Kilo Source Lines Of Code

1.3 Reference Documents

The indications provided in this document are based on the ones stated in the previous deliverables for the project, the RASD document [1], the DD document [2] and the ITPD document [3].

Moreover it is strictly based on the specifications concerning the RASD assignment [4] for the Software Engineering II project, part of the course held by professors Luca Mottola and Elisabetta Di Nitto at the Politecnico di Milano, A.Y. 2016/17.

To support the application of the COCOMO II model estimate, the COCOMO II Model Definition Manual [5] has been followed.

Section 2

Function Points Estimation

This chapter is devoted to the Function Point Analysis for the *PowerEnJoy* project, aimed at obtaining a reasonable estimate of the size of the project, which will be later used within the COCOMO II estimation model to compute an average effort factor for the development process.

2.1 Function Points Analysis

2.1.1 Internal Logic Files (ILFs)

Internal Logic Files are defined as follows [5]:

> *"Internal Logic Files count each major logical group of user data or control information in the software system as a logical internal file type. They include each logical file (e.g., each logical group of data) that is generated, used, or maintained by the software system."*

In practice, they can be identified as a homogeneous set of data used and managed by the application itself.

The identified ILFs for the application are:

ILF1 User data: identifies and groups all data pieces about users: information about user accounts, lists of user payments, positional information, reservation histories. The Function Point complexity can be set to *HIGH*, since user-related entities will be generated and used by the application in great number and the level of detail of said data pieces will be considerable: attributes for involved data structures will be many.

ILF2 Car data: identifies and groups all data about the application-managed vehicles: car status and availability, positional information. The number of instances related to this data group will be much more contained in number when compared with those of the **User data** group, and will also include a smaller number of attributes; hence, the complexity level can be set to *LOW*.

ILF3 Payment data: identifies data related to payments as simple payment characteristics. As this group of data is only maintained by the application and not used if not in combination with other data, the ILF is quite simpler than the two listed before. Moreover, the high number of potential instances will not increase the complexity, due to the very clean and simple description provided by the few possible data attributes, and thus the Function Point complexity will be set to *LOW*.

ILF4 Reservation data: identifies and groups all information related to reservations: reservation times and details, reserved car information, information about users performing actual reservations, information about eventual fees to be paid for reservation expiration. In this context, data is quite complex in structure, and the considerable number of attributes raises the complexity of the role of this ILF. Moreover, since the number of instances will probably be an order of magnitude greater than the user instances, the complexity of this Function Point will be set to *HIGH*.

ILF5 Ride data: identifies and groups all data involving rides: ride status and times, linked reservation information, information about the related payment, information detailing the eventual alternative charges situations detected during a ride. Similarly to what has been said about **Reservation data**, the great number of possible instances combines with the complexity of the potential data attributes and increases the overall complexity for the ILF. Hence, the Function Point complexity will be set as *HIGH* as well.

ILF6 Safe Area data: identifies a simple set of data involving the Safe Area: essentially boundaries and the positions of power grid stations. The number of potential instances for this data group is really low and stable, and the number of possible attributes is quite small. For these reasons, the Function Point complexity will be set to *LOW*.

2.1.2 External Interface Files (EIFs)

External Interface Files are defined as follows [5]:

> *"Files passed or shared between software systems should be counted as External Interface File types within each system."*

In practice, they can be identified as a homogeneous set of data used by the application, but generated and maintained by other applications.

The identified EIFs for the application are:

EIF1 Payment records data in the Payment Handlers databases: the data used by the application and managed by the Payment Handlers systems consist of the recorded transactions corresponding to the payments for using the services of *PowerEnJoy*. The complexity of said pieces of data is low, since the information is simple and must not be elaborated in any way to be utilized; data in this group is quite abundant, but being it so simple in structure the Function Point complexity will be classified as *LOW*.

EIF2 Intervention records in the Maintenance System database: As for the previous point, information about the maintenance interventions is quite simple and not must not be elaborated, but in this case it is not wrong to suppose that data in this group is quite scarce in number. Hence, the complexity of this Function Point will be set to *LOW* as well.

EIF3 Data streams related to the *Google Maps* service: data coming from the interaction with the external maps software is substantially different from the groups indicated in the previous points: information is often in need of pre-processing before becoming usable and, even when simple, usually more vast and complex if compared to what stated in the previous points. Moreover, the quantity of data exchanged is quite considerable; because of these reasons, the complexity of this Function Point is set to *AVERAGE*.

2.1.3 External Inputs (EIs)

External Inputs are defined as follows [5]:

> *"External Inputs count each unique user data or user control input type that enters the external boundary of the software system being measured."*

In practice, they can be identified as elementary operations to elaborate data coming from the external environment.

The identified EIs for the application are:

EI1 Registration procedure: registration procedures only consist of simple insertions of data elements related to a single ILF (**User data**). For this reason, the corresponding EI complexity will be set to *LOW*.

EI2 Login procedure: similarly, login procedure only imply simple reading operations, and these operations are still only related to the **User data** ILF. The complexity of this Function Point will hence be set to *LOW*.

EI3 Update procedure for user profiles: as already seen for the previous EIs, the procedures needed to update profile data and delete profiles make use of elementary update operations on data which is part of the **User data** ILF, and for this reason the associated complexity will be set to *LOW* as well; since the distinct operations for the FP are two, this will be counted twice.

EI4 Data streams from the sensors and equipment of cars: radically different are the operations related to the streams of information from sensors and vehicle equipment, representing useful data to be processed by the application. Said information is associated with the car availability updates, with the car monitoring, with the detection of alternative charges situations. Since the referred ILFs are many (**Ride data**, **Safe Area data** and **Payment data**) and the variety of operations is actually wide, the complexity of the Function Point will be classified as *HIGH*.

EI5 Car reservation procedure: the procedure used to reserve cars implies different operations based on the conditions in which the input is provided by the user: in fact, the EI includes both the operations involved with a reservation in itself and those involving the release of an active one; in both cases, the necessary operations are not complex in any way since they only refer a couple of ILFs (**User data** and **Reservation data**) and include combinations of reading operations and insertions or reading operations and update ones. Based on these considerations, a reasonable classification for the complexity of this Function Point is *LOW*, and this will be counted as two FPs because of the distinct tasks involved with the operation.

EI6 Car unlocking procedure: similarly to what stated for the previous point, the car unlocking procedures must be subdivided in two different

operations: one that uses a vehicle-specific code and one that relies on the user's position. This procedure refers at least to three ILFs (**User data**, **Car data** and **Reservation data**): since, however, the two operations only imply simple reading and matching elementary operations, they will be classified as *LOW* in terms of complexity, and counted twice as for the previous EI.

EI7 User authentication procedure for the rides: the procedure is very similar to the one described in the previous point, but only two ILFs are referred (**User data** and **Reservation data**). In general, this kind of operation is quite simpler than the car unlocking one, since it does not imply and cross-matching between more than two data entities. For this reason, the complexity of this Function Point can be estimated as *LOW*.

2.1.4 External Outputs (EOs)

External Outputs are defined as follows [5]:

> *"External Outputs count each unique user data or control output type that leaves the external boundary of the software system being measured."*

In practice, they can be identified as elementary operations that generate data for the external environment.

The identified EOs for the application are:

EO1 E-mail notification procedure: the output procedure required to generate e-mail notification does not need great elaboration of data apart from the essential operations used to read the information related to the event that must trigger an e-mail notification. The involved ILFs are at most one or two (**Payment data** or **User data**), and for this reason the complexity of this Function Point will be set to *LOW*.

EO2 User notification procedure: general-purpose notifications are even simpler than what stated for the e-mail-based ones, since they correspond to errors or confirmations caused by failure or success of data detection/insertion/update/deletion. They however differ from the previous classified notifications since they can refer, in general, to almost every ILF defined in the dedicated section above. For this reason, the complexity can be set to *AVERAGE*.

EO3 Procedure for the retrieval of available cars: the procedure is used to perform matches between internal and external data, namely the information about cars and their availability and the address position information from *Google Maps*; since the operations themselves are not complex but imply reading data from both internal (ILFs: **Car data**, **User data**) and external (EIFs: **Google Maps**) data sources, the overall complexity of the Function Point can be estimated as *HIGH*.

EO4 Procedure for the retrieval of final charges: this last output procedure is based on the simple elaboration of pieces of data from the **Ride data** ILF, namely information about ride duration and alternative charges conditions. This procedure implies simple arithmetical operations combined with more complex operations involving data from internal sources, and will hence be classified as *AVERAGE* in terms of complexity.

2.1.5 External Inquiries (EQs)

External Inquiries are defined as follows [5]:

> *"External Inquiries count each unique input-output combination, where input causes and generates an immediate output."*

In practice, they can be identified as elementary operation that involve input and output, without significant elaboration of data from logic files.

The identified EQs for the application are:

EQ1 Visualization of user profile data by the user him/herself: this EQ only implies presentation of user data upon requests from users themselves. Since it only refers one ILF (**User data**), its complexity will be set to *LOW*.

EQ2 Visualization of the Safe Area boundaries/Power Grid Stations positions: the same considerations taken for the previous Function Point can be made for this one, too. The presentation involves data related to the **Safe Area data** ILF, and the complexity will be set to *LOW*; the FP will be counted twice here, because the two procedures for retrieving, respectively, boundaries and power grids information are distinct.

EQ3 Retrieval of users' positions: an analogous analysis can be performed for this last Function Point, since the only ILF involved is **User data**. Hence, the complexity level will be set once more to *LOW*.

EQ4 Visualization of reservation information: lastly, the same idea applies to the simple process of visualizing the informations about currently active reservations. Since, again, the involved ILF is only one (namely **Reservation data**), the complexity of the procedure will be set to *LOW*.

2.2 Function Points Weights

Given the Function Points computation of the previous sections, the analysis can continue with the final estimate of the number of *Unadjusted Function Points (UFPs)*, as shown Table 2.2.

The assignments of weights to the different types of Function Points, based on their individual complexity, follows Table 2.1.

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

Table 2.1: A summary of the association of different weights to the individual Function Point type. The weights differ based on the level of complexity of the single Function Point.

Type	Function Point	Complexity	Weight
ILFs	ILF1	High	15
	ILF2	Low	7
	ILF3	Low	7
	ILF4	High	15
	ILF5	High	15
	ILF6	Low	7
ILF Subtotal			**66**
EIFs	EIF1	Low	5
	EIF2	Low	5
	EIF3	Average	7
EIF Subtotal			**17**
EIs	EI1	Low	3
	EI2	Low	3
	EI3	Low	$2 \times 3 = 6$
	EI4	High	6
	EI5	Low	$2 \times 3 = 6$
	EI6	Low	$2 \times 3 = 6$
	EI7	Low	3
EI Subtotal			**33**
EOs	EO1	Low	4
	EO2	Average	5
	EO3	High	7
	EO4	Average	5
EO Subtotal			**21**
EQs	EQ1	Low	3
	EQ2	Low	$2 \times 3 = 6$
	EQ3	Low	3
	EQ4	Low	3
EQ Subtotal			**15**
Total UFPs			**152**

Table 2.2: Computed weights for all the detected function points.

2.3 Function Points Estimation Results

Based on the considered parameters, the final value for the UFPs of this project is: **152**.

UFPs are used to estimate the Lines of Code of a software project depend-

ing on the average number of SLOC per FP for a given language. Assuming JEE will be the programming language taken into account to develop this project, the average multiplicative factor is **46** [7]; an upper-bound to the estimate can be calculated using an appropriate multiplicative factor, which for JEE is defined as **67** [7].

The final computation for the average number of SLOC for the project is:

$$\#SLOC := UFPs \times LDF_{avg} = 152 \times 46 = 6992 \textbf{ SLOC} \tag{2.1}$$

The value of the mentioned upper-bound for the number of SLOC estimate is:

$$\#SLOC := UFPs \times LDF_{high} = 152 \times 67 = 10184 \textbf{ SLOC} \tag{2.2}$$

Section 3

COCOMO II Effort Estimation

3.1 Scale Drivers

Some of the most important factors contributing to the duration and cost of a project are the Scale Drivers. Each of said drivers describes the project itself and determines the exponent used in the Effort Equation. More precisely, the Scale Drivers reflect the non-linearity of the effort with relation to the number of SLOC. Each scale driver has a range of rating levels from *Very Low* to *Extra High*. The result of the evaluation is summed up in Table 3.1.

Precedentedness: points out the previous experience of the team with the organization and development of large scale projects. Since the team members are new to most of the notions concerning this kind of projects, the Precedentedness is *LOW*.

Development Flexibility: reflects the level of flexibility in the development process with respect to the given specifications and requirements. Since the project is bounded to a set of prescribed specifications, but a certain degree of flexibility is allowed in the definition of the requirements, the Development Flexibility is *NOMINAL*.

Risk Resolution: reflects the level of awareness and responsiveness with respect to risks. A rather detailed risk analysis, followed by possible countermeasures, is offered in Section 5. Therefore, the Risk Resolution driver is set to *HIGH*.

Team Cohesion: indicates how good the relationship among the team members is and how well the development team works together. Since the team is effective and cohesive with no communication problems, the parameter is set to *VERY HIGH*.

Process Maturity: by means of a set of levels, it describes how well the behaviours, practices and processes of an organization can reliably produce required outcomes. It is set to *CMM Level 3,* which corresponds to a defined level with respect to the software development process of our organization. At this level the organization has developed its own standard software process and has a greater attention to documentation, standardization and integration. In the COCOMO II model, this corresponds to a level of *HIGH*.

Scale Driver	Factor	Value
Precedentedness (PREC)	Low	4.96
Development Flexibility (FLEX)	Nominal	3.04
Risk Resolution (RESL)	High	2.83
Team Cohesion (TEAM)	Very High	1.10
Process Maturity (PMAT)	High	3.12
Total	$E = 0.91 + 0.01 \times \sum_i SF_i$	1.0605

Table 3.1: Result of the scale drivers analysis.

3.2 Cost Drivers

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

Since this effort analysis is carried out at the beginning of the life-cycle of the project, before the writing of the RASD [1] and the DD [2], the point-of-view will be that of the *early design* version of the COCOMO II model. Clear information about the architecture are still not available and the team is exploring and evaluating different architectural alternatives.

In order to conduct the analysis of said cost drivers, guidelines given by the COCOMO Model Definition Manual [5] have been followed.

Each cost driver of the post-architecture approach has a rating that is considered as a numerical value during the early design analysis (Very Low = 1, Extra High = 6). Ratings related to the Post-Architecture cost drivers corresponding to an Early Design one will be summed and converted into rating levels using the provided tables.

Personnel Capability: describes the overall capabilities of the team members in terms of problem-solving, actual implementation skills and rat-

ings of personnel turnover. It derives from the conjunction of the Analyst Capability (ACAP) factor, the Programmer Capability (PCAP) factor and the Personnel Continuity (PCON) factor. In detail, the ACAP factor expresses the analysis and design ability of the team members, as well as the ability to communicate and cooperate. It is set to Nominal (3) since the team members have no previous experience in finding requirements, but communication and cooperation is very high. PCAP is set to Nominal (3) too, because the programming abilities of the team is in the average. PCON is instead set to Very High (5) since the turnover is totally absent.

The PERS factor is set to High according to the following table:

Sum of ACAP, PCAP, PCON Ratings	3,4	5,6	7,8	9	10,11	12,13	14,15
Rating Levels	Extra Low	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	2.12	1.62	1.26	1.00	0.83	0.63	0.50

Table 3.2: PERS cost driver.

Product Reliability and Complexity: this Early Design cost driver depends on the characteristics of the product under development and combines the following four Post-Architecture cost drivers: Required Software Reliability (RELY), Database Size (DATA), Product Complexity (CPLX), and Documentation Match to Life-cycle Needs (DOCU). RELY is set to Nominal (3) since a software failure brings moderate financial losses, but there is no risk to human life. DATA instead is set to Very High (5) since the size of the testing database can be estimated to be 100MB while the number of SLOC has been computed in the Section about FPs. All this leads to a D/P parameter (Test Database Size over SLOC ratio) that is grater than 1000 [**?**]. CPLX is set to Nominal (3) because no complex algorithm are involved and I/O processing includes status checking and error processing. Moreover, the user interface will be of a medium complexity as no complicated widgets are planned to be used. DOCU is set to Nominal (3) too, since the team does not want to spend excessive time on documentation, but a right-sized one is very important to reduce system maintenance costs and improve the software understanding.

The RCPX factor is set to High according to the following table:

Sum of RELY, DATA, CPLX, DOCU Ratings	5,6	7,8	9-11	12	12-15	16-18	19-21
Rating Levels	Extra Low	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.49	0.60	0.83	1.00	1.33	1.91	2.72

Table 3.3: RCPX cost driver.

Re-usability: this cost driver points out the additional effort needed to construct components intended to be reused on current or future projects. A Nominal value is set because re-usability is actually limited to the project itself and no similar projects are planned for the future.

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

Table 3.4: RUSE cost driver.

Platform Difficulty: the three Post-Architecture cost drivers TIME, STOR and PVOL are here combined together to express the intrinsic complexity of the platform. TIME is a measure of the execution time constraints imposed upon a software system. Since *PowerEnJoy* will offer a service that is expected to be extensively and frequently used by a large customer base, TIME is set to High (4). STOR represents the usage of the available storage offered by the system. As said before, the service provided by the system is supposed to be repeatedly and regularly used. On the other hand, modern technologies can offer a lot of storage capacity in order to prevent saturation, therefore STORE is set to High (4). The PVOL driver expresses the platform volatility. The system will not require frequent major updates, but minor changes, patches and bug fixes are planned to be release constantly in time. Therefore, PVOL is set to Low (2).

The PDIF factor is set to High according to the following table:

Sum of TIME, STOR, PVOL Ratings	8	9	10-12	13-15	16-17
Rating Levels	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.87	1.00	1.29	1.81	2.61

Table 3.5: PDIF cost driver.

Personnel Experience: combines the three Post-Architecture cost drivers APEX, PLEX and LTEX. APEX expresses the level of experience of the project development team. Since the team members have some experience with Java, but never used JEE, the driver is set to Low (2). PLEX describes the productivity influence of the platform experience. As said above, the development team has no experience with technologies supporting JEE, but has some experience with databases and networking. Therefore, PLEX is set to Low (2). LTEX is set to Low (2) too, for the same reasons expressed above, but it must be pointed out that the team members have some basic experience of tools aimed to support requirements modelling and analysis.

The PREX factor is set to Very Low according to the following table:

Sum of APEX, PLEX, LTEX Ratings	3,4	5,6	7,8	9	10,11	12,13	14,15
Rating Levels	Extra Low	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.59	1.33	1.22	1.00	0.87	0.74	0.62

Table 3.6: PREX cost driver.

Facilities: the use of software tools (TOOL) and the multi-site development (SITE) drivers are combined together to express how the infrastructure and tools influences the project. Software tools that will be used in the project are capable, mature, integrated and able to manage the life cycle of the software-to-be, therefore the rating level is set to High (4). The team members usually collaborate residing in the same city. Moreover, internet services (chat, email, video conference...) are frequently used, therefore SITE is set to Very High (5).

The FCIL factor is set to Very High according to the following table:

Sum of TOOL, SITE Ratings	2	3	4,5	6	7,8	9,10	11
Rating Levels	Extra Low	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.43	1.30	1.10	1.00	0.87	0.73	0.62

Table 3.7: FCIL cost driver.

Required Development Schedule: measures the schedule constraints imposed on the project team developing the system-to-be. It is set to Nominal since deadlines are not so flexible and effort must be equally distributed over the given development time.

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

Table 3.8: SCED cost driver.

The overall results of the analysis of the cost drivers are summarized by the following table:

Cost Driver	Factor	Value
Personnel Capability (PERS)	High	0.83
Reliability and Complexity (RCPX)	High	1.33
Re-usability (RUSE)	Nominal	1.00
Platform Difficulty (PDIF)	High	1.29
Personnel Experience (PREX)	Very Low	1.33
Facilities (FCIL)	High	0.73
Required Development Schedule (SCED)	Nominal	1.00
Total	$EAF = \prod_i C_i$	1.3826

3.3 Effort Equation

Equation 3.1 allows to estimate the effort in Person-Months (PM):

$$\text{Effort} = A \times EAF \times KSLOC^E \tag{3.1}$$

Where the parameters have the following meanings:

- $EAF = \prod_i C_i$: derived from the cost drivers analysis.
- $E = 0.91 + 0.01 \times \prod_i SF_i$: derived from the scale drives analysis.
- $KSLOC$: Kilo Source Lines of Code estimated via FPs analysis.
- $A = 2.94$

The total effort results in **31.97** person-months using the value obtained with 2.1; the result is of **47.64** person-months in case the upper-bound estimate 2.2 is used.

3.4 Schedule Estimation

The duration of the project is calculated using Equation 3.2.

$$\text{Duration} = 3.67 \times (PM)^{0.28+0.2\times(E-B)} \tag{3.2}$$

where B is equal to 0.91 for COCOMO II and PM is the effort calculated by Equation 3.1. The exponent E depends on the scale drivers analysed above.

Using the value of PM obtained with the average estimate of SLOC, the duration results in 10.747 months, corresponding to 2.97 developers. Using the value of PM obtained with the upper-bound estimate of SLOC, the duration results in 12.162 months, corresponding to 3.9 developers.

Since our team is composed only by two people, the estimated development time would be:

$$\frac{31.97 \text{ pm}}{2 \text{ people}} = 15.9 \text{ months}$$

or, as upper-bound estimate:

$$\frac{47.64 \text{ pm}}{2 \text{ people}} = 23.8 \text{ months}$$

so, the project overall time will be around 16 months, and most probably not greater than 24 months.

Section 4

Schedule Planning and Resource Allocation

4.1 Tasks and Schedule

The main tasks of which the *PowerEnJoy* project is composed of are the following:

1. **Requirements Analysis and Specification Document (RASD)** delivery - deliver a document containing the description of all goals, domain assumptions, functional and non-functional requirements for the project;

2. **Design Document (DD)** delivery - deliver a document describing the architectural design of the software system to be produced;

3. **Integration Test Plan Document (ITPD)** delivery - deliver a document containing the strategy to perform integration testing on the components of the system;

4. **Project Plan Document (PPD)** delivery - deliver a document containing the description of the schedule and tasks for the project and an estimation of the effort and size of the project itself, as well as an analysis of the risks that the project could face during its life-cycle;

5. **Implementation** - implement the software product and thoroughly write unit tests for all the code;

6. **Integration testing** - test the integration of all the software components of the project;

During the project life-cycle, some of these task can not begin if others are not completed yet. To illustrate the precedence constraints of the case, a **dependency graph** is provided in Figure 4.1.

Note that, however, since a software project is highly subject to change in requirements and evolves continuously, there might be the need of coming back to previous tasks one or more times after the conclusion of said tasks themselves.

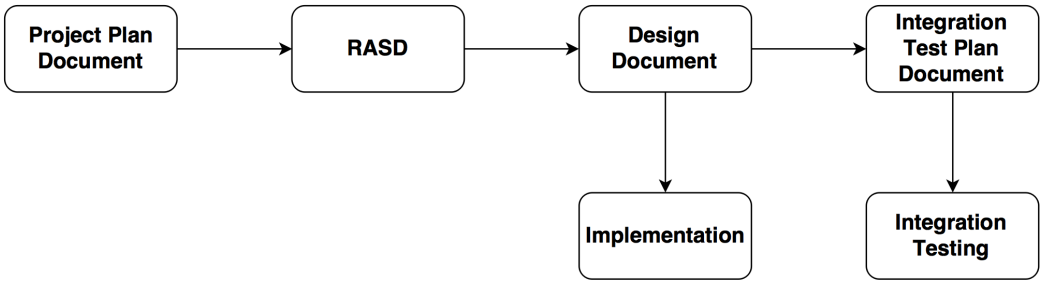

Figure 4.1: Dependency graph illustrating precedence constraints.

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

4.2 Resource Allocation

Given the tasks defined above, the allocation of human resources of the team to the single tasks and their sub-parts is going to be defined in this section. Since the team is only composed of two members, the granularity of the activities composing the tasks is quite rough: this is to avoid a pointless level of detail in favour of an easier-to-understand allocation of macro-activities.

Note that most activities involve contribution from both team members, either in a parallel fashion or in a cooperative way. This is done in order to increase the awareness of the team with respect to the single tasks of the project, so that all the members can be work on any part of the system in case of necessity, without spending too much time understanding it; this also reduces the risk of misunderstandings between team members and increases the efficiency of cooperative work sessions.

The following tables describe in detail the subdivision of every task in different activities, as well as the allocation of team members to the single activities themselves. The duration of tasks and activities is structured based on the considerations and conclusions expressed in the sections above.

Activities on the same row represent activities executed in parallel. If the same activity is marked twice on the same row, that activity is executed in cooperation by both team members.

	Giovanni	**Marco**
1_{st} **week**	Risk Management	Introduction
	FPs Estimation	FPs Estimation
	COCOMO II Effort Estimation	COCOMO II Effort Estimation
	Resource Allocation	Resource Allocation
	Review Resource Allocation	Review Risk Management

Table 4.1: Resource allocation for Project Plan Document.

	Giovanni	**Marco**
1_{st} **week**	Purpose, Scope	References, Overview
	Initial goals draft	Initial goal draft
	Product Perspective	Product Functions, Constraints
	User Characteristics	Constraints
2_{nd} **week**	Initial assumptions draft	Initial assumptions draft
	Goals	Assumptions and Dependencies
3_{rd} **week**	Initial requirements draft	Initial requirements draft
	Initial use-case draft	Refinement of goals and assumptions
	Specific Requirements	Final writing of Overall Description
	Specific Requirements	Specific Requirements
4_{th} **week**	World and Machine diagram	World and Machine diagram
	Appendix	Bibliography
	Final revision	Final revision

Table 4.2: Resource allocation for RASD.

	Giovanni	Marco
1_{st} **week**	Introduction	High level draft of architecture
	Identification of main components	Identification of main components
	Overview	High Level Components
2_{nd} **week**	First draft of component interfaces	First draft of component interfaces
	Component View	Deployment View
	Runtime View	Runtime View
	Component Interfaces	Component Interfaces
3_{rd} **week**	Algorithm Design	Final writing of Architectural Design
	UI mock-ups	UI mock-ups
	User Interface Design	Requirements Traceability
4_{th} **week**	Bibliography	Appendix
	Final revision	Final revision

Table 4.3: Resource allocation for Design Document.

	Giovanni	Marco
1_{st} **week**	First Integration Strategy draft	Introduction
	Integration Strategy	Integration Strategy
2_{nd} **week**	Individual Steps and Test Description	Individual Steps and Test Description
	Test cases revision	Test cases revision
	Program Stubs required	Test Data required
3_{rd} **week**	Bibliography	Tools and Test Equipment
	Final Revision	Appendix

Table 4.4: Resource allocation for ITPD.

Section 5

Risk Management

The *PowerEnJoy* system project might be threatened by many risks. The analysis of said risks follows a specific approach that splits them up into three groups: *project*, *technical* and *business risks*.

In the following sections a detailed description of the risks and the countermeasures is provided. Moreover, summarizing tables at the end of each section give an overview of the probability and the effects of each of the listed risks. The probability of a risk occurring can be classified as: *Very Low*, *Low*, *Moderate*, *High*, *Very High*, *Certain*. The effects of a risk upon occurrence can be classified as: *Negligible*, *Moderate*, *Serious*, *Catastrophic*.

5.1 Project Risks

Project risks pose a threat to the project plan and make the project schedule slip, increasing the overall costs. The following risks have been found and analysed, providing a response strategy.

Changing requirements: during the development of the project the requirements can change unexpectedly. This kind of risk cannot be prevented, but it can be reduced by using a style of programming that takes advantage of reusable and extensible code.

Deadlines not met: it can occur that the project requires more time than expected to be carried out. In this case only some functionalities will be offered in a first release while less essential features will be developed later. The above-mentioned first release can provide the target services by the only means of the Mobile Application. The Web Application and the Web Tier may be built afterwards.

Lack of communication: team members usually work separately. This can cause misunderstandings and conflicts about the division of different tasks. To mitigate the risk, crystal clear and complete specification and design documents must be provided. Moreover the responsibilities of each group member must be clearly defined in this document. Lastly, a good countermeasure is to arrange frequent brief meetings in order for each member to gain awareness of the level of progress achieved.

Team break-ups: if, for any natural or human reason, any of the team member is forced to leave the project team, the development process will inevitably suffer from huge delays and the risk will result in an impossibility to meet the defined deadlines (this is due to the very small size of the team). This can be countered, in case of social issues, in a pre-emptive way by implying a work method that values each team member in the same way; in case of natural causes, the reactive behaviour to be adopted must be that of hiring new members. Note that, in this second unpredictable case, the time required by the project will be greatly increased also due to the well-acknowledged Brooks's Law [6].

Risk	Probability	Effects
Changing requirements	Moderate	Moderate
Deadlines not met	Moderate-High	Moderate
Lack of communication	Very Low	Negligible
Team breaks-up	Very Low	Catastrophic

Table 5.1: Evaluation of project risks.

5.2 Technical Risks

Technical risks threaten the quality and the punctuality of the software product preventing a straightforward implementation. The following risks have been found and analysed, providing a response strategy.

Unreadable code: large projects may have a badly structured and unreadable code. Documenting the code can be a reasonable way to mitigate this risk. Furthermore, information provided by a good Design Document can come in handy too.

Scalability issues: if the system does not scale properly with the increasing number of users, a work of major redesign will be carried out. To reduce

the risk, a correct estimation of the computing power and the number of involved machines is necessary.

Integration testing failure: in case, during the integration testing phase, the team realizes that the system components do not integrate as they should based on the tests defined in the ITPD [3], there can be the risk of delays in the overall project schedule, resulting in a fail in meeting deadlines. In case this happens, the efforts of the team members will be fully redirected to redefining and implementing integration test cases over other tasks in progress.

Downtime: Upon being fully functional and operative, the system may experience brief or long periods of downtime. This will be accounted for and discussed in other documents, and the risk could be made less likely by decoupling the clients' activities and making some clients independent from other components of the system, so that a channel to access the application is always active while the other is being fixed. In cases in which this is not possible, structuring the code to make it more reliable, robust and maintainable is always a good countermeasure.

Data loss and leaks: if, for any reason, a big portion of the application data should be lost, the damage to the application itself could be considerable. For this reason, it is recommendable to store data in multiple locations, or to have one or more backup database, especially for sensitive information.

Interaction with external systems: since the system heavily relies on the interaction with payment handlers and a maintenance system for interventions on vehicles, failures involving components devoted to the communication with said external systems constitute a severe risk. A possible countermeasure is to maintain a close collaboration with development teams of the partners.

Risk	Probability	Effects
Unreadable code	Moderate	Moderate
Scalability issues	Low	Serious
Integration testing failure	Moderate	Serious
Downtime	Low	Moderate
Data loss and leaks	Low	Catastrophic
Interaction with external systems	Low	Serious

Table 5.2: Evaluation of technical risks.

5.3 Economical Risks

Economical risks jeopardize the whole software product threatening its viability. The following risks have been found and analysed, providing a response strategy.

Bankruptcy: the income from the usage of the application may not be sufficient to support maintenance and development of the system. A good feasibility study should help avoiding this critical situation.

Local regulation and policies: minor risks could derive from changes in the local regulation about traffic within the city, and especially with respect to the areas that will be covered by the service. The minor problems deriving from this can be mitigated by maintaining a continuous dialogue with the local administration for the city (potentially, cities) in which the *PowerEnJoy* service will be deployed.

Competitors: competition by other providers of the same kind of services must be kept in consideration at all times, since a poor management could lead considerable economical losses. Competition must be exploited to the advantage of the *PowerEnJoy* company by continuously providing new appealing functionalities, based in part on customer feedback.

Risk	Probability	Effects
Bankruptcy	Moderate	Catastrophic
Local regulation and policies	Very Low	Negligible
Competitors	High	Moderate

Table 5.3: Evaluation of economical risks.

Appendix A

Appendix

A.1 Software and tools used

- LaTeX, used as typesetting system to build this document.
- draw.io - `https://www.draw.io` - used to draw diagrams and mock-ups.
- GitHub - `https://github.com` - used to manage the different versions of the document and to make the distributed work much easier.
- GitHub Desktop, the GitHub official application that offers a seamless way to contribute to projects.

A.2 Hours of work

The absolute major part of the document was produced in group work. The approximate number of hours of work for each member of the group is the following:

- Giovanni Scotti:
- Marco Trabucchi:

NOTE: indicated hours include the time spent in group work.

Bibliography

[1] AA 2016/2017 Software Engineering 2 - *Requirements Analysis and Specification Document* - Giovanni Scotti, Marco Trabucchi

[2] AA 2016/2017 Software Engineering 2 - *Design Document* - Giovanni Scotti, Marco Trabucchi

[3] AA 2016/2017 Software Engineering 2 - *Integration Test Plan Document* - Giovanni Scotti, Marco Trabucchi

[4] AA 2016/2017 Software Engineering 2 - *Project goal, schedule and rules*

[5] 1995 - 2000 Center For Software Engineering, USC - *COCOMO II - Model Definition Manual* - Version 2.1

[6] 1975, *Frederick P. Brooks Jr.*, The Mythical Man-Month, Addison-Wesley

[7] Quantitative Software Management Website - *Function Point Languages Table* - `http://www.qsm.com/resources/function-point-languages-table`