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

RASD: Requirements Analysis and Specification Document

DD: Design Document

ITPD: Integration Test Plan Document

FP: Function Point

FPA: Function Point Analysis

COCOMO: COnstructive COst MOdel

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

ILF: Internal Logic Files

EIF: External Interface Files

EI: External Inputs

EO: External Outputs

EQ: External Inquiries

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. Although the data might appear complex in structure, the straightforward nature of most of its not-so-many attributes decreases its overall complexity. Since, however, 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 *AVERAGE*.

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 is balanced by the simple structure of the potential data attributes. Hence, the Function Point complexity will be set as *AVERAGE*.

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 not aggregated; 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 aggregated, but in this case it is not wrong to suppose that data in this group in quite scarce. Hence, the complexity of this Function Point will be set to *LOW* as well.

EIF3 The 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 aggregated and, even when simple, usually more vast and complex. 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 The 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 The 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 The update procedure for user profiles: as already seen for the previous EIs, the procedures needed to update profile data 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.

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 The 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: the regular reservation procedure and the alternative procedure in case a reservation is already active; both 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 *AVERAGE*.

EI6 The car unlocking procedure: although it only implies basic read-and-compare sequences of operations in order to check the conformity of users and the reservation information about who reserved a car, the car unlocking procedure refers at least to three ILFs (**User data**, **Car data** and **Reservation data**): for this reason it will be classified as *AVERAGE* in terms of complexity.

EI7 The 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 The 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 The 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 The 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 and external data source, the overall complexity of the Function Point can be estimated as *AVERAGE*.

EO4 The 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 only implies simple arithmetical operations, and will hence be classified as *LOW* 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 The 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 The 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*.

2.2 Function Points Weights

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

The assignments of weights to the different types of Function Points, based on their individual complexity, follows the table below:

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.

2.3 Function Points Estimation Results

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

Section 3

COCOMO II Effort Estimation

3.1 Scale Drivers

Some of the most important factors contributing to a project's duration and cost are the Scale Drivers. Each of the 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 not still 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).

Personnel Capability: describes the overall capabilities of the team members in terms of problem-solving, actual implementation skills and ratings 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 details, 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 not previous experience in finding requirements, but communication and cooperation is very high. Also PCAP is set to Nominal (3) because the programming abilities of the team is in the average. PCON is instead set at 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.

Re-usability: this cost driver points out the additional effort needed to construct components intended for reuse 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 3.3: RUSE cost driver.

Platform Difficulty:

Personnel Experience:

Facilities:

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 3.4: 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)		
Reliability and Complexity (RCPX)		
Re-usability (RUSE)		
Platform Difficulty (PDIF)		
Personnel Experience (PREX)		
Facilities (FCIL)		
Required Development Schedule (SCED)		
Total	$EAF = \prod_i C_i$	

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.
- $A = 2.94$

3.4 Schedule Estimation

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.

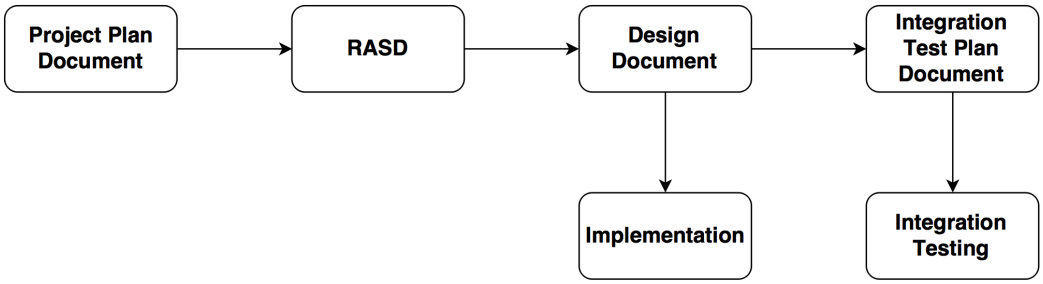

Figure 4.1: Dependency graph illustrating precedence constraints.

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**	ciao	ciao
	ciao	ciao
	ciao	ciao
	ciao	ciao
2_{nd} **week**	ciao	ciao
	ciao	ciao
	ciao	ciao
	ciao	ciao
3_{rd} **week**	ciao	ciao
	ciao	ciao
	ciao	ciao
	ciao	ciao
4_{th} **week**	ciao	ciao
	ciao	ciao
	ciao	ciao
	ciao	ciao

Table 4.2: Resource allocation for RASD.

	Giovanni	Marco
1_{st} **week**	ciao	ciao
	ciao	ciao
	ciao	ciao
	ciao	ciao
2_{nd} **week**	ciao	ciao
	ciao	ciao
	ciao	ciao
	ciao	ciao
3_{rd} **week**	ciao	ciao
	ciao	ciao
	ciao	ciao
	ciao	ciao
4_{th} **week**	ciao	ciao
	ciao	ciao
	ciao	ciao
	ciao	ciao

Table 4.3: Resource allocation for Design Document.

	Giovanni	Marco
1_{st} **week**	ciao	ciao
	ciao	ciao
	ciao	ciao
	ciao	ciao
2_{nd} **week**	ciao	ciao
	ciao	ciao
	ciao	ciao
	ciao	ciao
3_{rd} **week**	ciao	ciao
	ciao	ciao
	ciao	ciao
	ciao	ciao

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