POLITECNICO DI MILANO
2016-2017

Software Engineering 2: PowerEnJoy

Project Plan
Version 1.0

Peverelli Francesco
Reppucci Federico

TABLE OF CONTENT

1. INTRODUCTION

1.1. REVISION HISTORY

- 22.01.2017 release of the document

1.2. PURPOSE AND SCOPE

The PowerEnJoy project aims to develop a car-sharing service run exclusively employing electric cars. The system will provide a mobile application by means of which the users, once registered, will be able to use the car sharing services. The main goals of the service are to provide a sustainable and environmentally-friendly car sharing service as well as to promote virtuous behaviors from its users.

The purpose of the Project Plan document is to generate an estimate, as accurate as possible, of the time and effort needed to complete the project. In this document we apply well known models such as the function points analysis and COCOMO II, and the obtained results are used to create a schedule for the project and allocate the available resources. Furthermore, a risk management plan is devised.

1.3. DEFINITIONS AND ACRONYMS

ILF: Internal Logical File

EIF: External Interface File

EO: External Output

EI: External Input

EQ: External Inquiry

DET: Data Element Type

RET: Record Element Type

FTR: File Types Referenced

UFP: Unadjusted Function Point

PREC: Precedentedness

FLEX: Development Flexibility

RESL: Risk Resolution

TEAM: Team Cohesion

PMAT: Process Maturity

RELY: Required Software Reliability

DATA: Database Size

CPLX: Product Complexity

RUSE: Developed for Reusability

DOCU: Documentation match to Life-cycle needs

TIME: Execution Time Constraint

STOR: Main Storage Constraint

PVOL: Platform Volatility

ACAP: Analyst Capability

PCAP: Programmer Capability

PCON: Personnel Continuity

APEX: Applications Experience

LTEX: Language and Tool Experience

TOOL: Use of Software Tools

SITE: Multisite Development

SCED: Required Development Schedule

1.4. REFERENCE DOCUMENTS

- The PowerEnJoy Requirements Analysis Specification Document (RASD)
- The PowerEnJoy Design Document (DD)
- The PowerEnJoy Integration Test Plan Document (ITPD)
- http://www.functionpointmodeler.com/fpm-infocenter/index.jsp

2. COST ESTIMATION

2.1. FUNCTION POINTS

2.1.1. FUNCTION POINTS METHODOLOGY

The main purpose for this function points analysis is to estimate the size of the application, in order to come up with an accurate prediction of the time and resources required to complete the development process.

The function points methodology categorizes data at rest and data in motion in order to derive the lines of source code required to develop the corresponding functionalities, scaled based on a multiplicative factor depending on the programming language used.

The chart reported here specifies the multiplicative factor used for each programming language or framework involved

Java	53
Objective C	27
C#	54
J2EE	46

2.1.2. FUNCTION POINTS CALCULATION

INTERNAL LOGICAL FILES

Any set of data originated and maintained by the application is considered an Internal Logical File (ILF).

ILFs are rated and scored, and the rating is based upon the number of data elements (DETs) and the record element types (RETs).

DET's are atomic data visible by the user. Record element types have a more elusive definition, but in simple terms can be seen as the number of sub-records contained within the main record of each ILF.

The table below lists both the level (low, average or high) and appropriate score (7, 10 or 15).

In the following paragraph we list the ILFs identified within our application and their corresponding rating:

User data: record storing information about the users

DETs: 18

RETs: 3

Corresponding complexity: Low

Reservation: record storing information about a user-made reservation

DETs: 5

RETs: 2

Corresponding complexity: Low

Profile stats: data derived from each user's rides

DETs: 3

RETs: 1

Corresponding complexity: Low

Payment: record of a user's payment

DETs: 9

RETs: 3

Corresponding complexity: Low

Ride: record containing information about a single ride

DETs: 5

RETs: 3

Corresponding complexity: Low

Safe area: a record storing the information about safe areas

DETs: 3

RETs: 1

Corresponding complexity: Low

Car state: record storing information about the sate of a car

DETs: 9

RETs: 1

Corresponding complexity: Low

Retrieval request: record storing a car retrieval request

DETs: 5

RETs: 2

Corresponding complexity: Low

Employee data: record storing information about an employee

DETs: 6

RETs: 2

Corresponding complexity: Low

EXTERNAL INTERFACE FILES

External Interface Files (EIF) are files created and managed by another application, but used by the application for which the function point count is being carried out.

The rating is based upon the number of data elements (DETs) and the record types (RETs). The table below lists both the level (low, average or high) and appropriate score (5, 7 or 10)

Maps provided by Google

Considering that the maps are provided by an external service, an accurate analysis of DETs RETs proves to be rather impractical, but we can assume the worst-case amount of data to manage to be high.

Corresponding complexity: High

UFP = 1 * 10 = 10

EXTERNAL INPUT

External Inputs (EI) are transactions (or elementary processes) that bring data from outside the application domain (or application boundary) to inside that application boundary.

EI's are rated and scored according to the number of data element types (DET's) and the file types referenced (FTR's).

File Types Referenced(FTR) are file types referenced by a transaction. An FTR must also be an internal logical file or external interface file.

The table below lists both the level (low, average or high) and appropriate score (3, 4 or 6).

Registration (with data insertion)

DETs: 18

FTRs: 1

Corresponding complexity: average

Car reservation

DETs: 5

FTRs: 3

Corresponding complexity: high

Modify profile data

DETs: 17

FTRs: 1

Corresponding complexity: average

NOTE: the function point count takes into consideration the fact that three different clients will be developed through three different programming languages. As a result of this, each input, output or inquiry involving both the client side and the server of the application will be counted in total four times, each time with the appropriate coefficient depending on the selected programming language

UFP = 2 * 4 + 6 = 14 (considered four times)

EXTERNAL OUTPUT

External Outputs (EO) are transactions (or elementary processes) that take data from a resting position to outside the application domain and present information to a user through processing logic other than, or in addition to, the retrieval of data or control information. The processing logic must contain at least one mathematical formula or calculation, create derived data, maintain one or more ILFs or alter the behavior of the system.

EOs are rated and scored according to the number of data elements (DETs) and the file types referenced (FTRs). The table below lists both the level (low, average or high) and appropriate score (4, 5 or 7).

Retrieval notification

DETs: 2

FTRs: 1

Corresponding complexity: low

Display personal data

DETs: 19

FTRs: 1

Corresponding complexity: low

Display payment history

DETs: 9

FTRs: 1

Corresponding complexity: low

UFP = 2 * 4 (considered four times) + 4 = 8 (considered four times) + 4

EXTERNAL INQUIRY

External Inquiry (EQ) is an elementary process with both input and output components that results in data retrieval from one or more internal logical files and external interface files. The input process does not update or maintain any FTRs (Internal Logical Files or External Interface Files) and the output side does not contain derived data.

The rating is based upon the total number of unique (combined unique input and out sides) data elements (DETs) and the file types referenced (FTRs). An EQ is rated (Low, Average or High) like an EO, but assigned a value like and EI.

Payment data validation: validation of the credit card data provided by the user

DETs: 2

FTRs: 1

Corresponding complexity: low

Password retrieval

DETs: 2

FTRs: 1

Corresponding complexity: low

Payment request

DETs: 3

FTRs: 1

Corresponding complexity: low

Browse website (request of the static page)

DETs: 1

FTRs: 1

Corresponding complexity: low

Login

DETs: 2

FTRs: 1

Corresponding complexity: low

Car positions inquiry

DETs: 4

FTRs: 1

Corresponding complexity: low

Map inquiry (to the external system)

DETs: 2

FTRs: NA

Corresponding complexity: low

UFP = 6 * 3 (considered four times) + 3 = 18 (considered four times) + 3

TOTAL COUNT = 80 + 40 (considered four times)

SLOC count: 46 * 80 + 40 * (53 + 27+ 54+ 46) = 10.880 SLOC

2.2. COCOMO II

2.2.1. COCOMO II METODOLOGY

The COnstructive COst MOdel is an open model based on a study of thousands of software projects, which aims to deliver an accurate estimation of the Person-Months required for the completion of a given software product, evaluated through the Effort Equation. This formula takes into account different parameters which depend on the characteristics of the project as a whole as well as the teams supposed to develop it. We chose to use the post architecture version of the model, since it provides a more accurate analysis.

2.2.2. COCOMO II CALCULATION

SCALE FACTOR

Precedentedness (PREC)

[VERY LOW - 0,05]

This is the first project for the development team with this characteristics and size.

Development Flexibility (FLEX)

[NOMINAL - 0,03]

The only pre-established requirements are the pre-existing interfaces of the cars

Architecture / Risk Resolution (RESL)

[Nominal - 4,24]

Team Cohesion (TEAM)

[VERY HIGH - 0,01]

The development team is composed by 2 student attending a MSc Computer Engineering. The communication is agile and efficient since they are colleagues that work together every day, and are on good terms with each other. Furthermore, they have worked together in previous projects.

Process Maturity (PMAT)

[VERY LOW - 0,05]

The development process is unstructured and unsupervised. The division of labor is weekly established and the stakeholders are not involved in our decisions.

POST-ARCHITECTURE EFFORT ESTIMATION

Product factors

Required Software Reliability (RELY)

[NOMINAL - 1,00]

A system failure can at most cause the loss of a small amount of money. For instance, during the failure, a user could not be able to reserve a car or the payment procedure could not be well performed.

Database Size(DATA)

[NOMINAL - 1,00]

The DATA cost driver is obtained by calculating the ratio of bytes in the testing database to SLOC in the program.

In the testing database, we assume that each field, being always raw data (string, dates, identifiers, numbers, Booleans), can take up at most 20byte. This is a pessimistic upper bound, useful for a quick estimate of the testing database size.

We consider to have:

- 100 users: each user, considering all its fields, takes up at most 20byte * 18 = 360byte
- 10 ride/user: each ride, considering all its fields, takes up at most 20byte * 5 = 100byte
- 10 reservation/user: each reservation, considering all its fields, takes up at most 20byte * 5 = 100byte
- 20 cars: each car, considering all its fields, takes up at most 20byte * 9 = 180byte
- 20 retrieval requests: each retrieval requests, considering all its fields, takes up at most 20byte * 5 = 100byte

- 20 employee: each employee, considering all its fields, takes up at most 20byte * 6 = 120byte

- 1000 payments: each payment, considering all its fields, takes up at most 20byte * 9 = 180byte

- 100 profile stats: each profile stats, considering all its fields, takes up at most 20byte * 3 = 60byte

The safe area is mainly a collection of points. Each point has 2 coordinates. Each coordinate we can suppose takes up 8byte.

We consider to have 100 points the describe the edges of polygons: the total occupied size is 100 * 8 * 2 byte = 16000byte

The total size is: 446KB

From the functional point, the SLOC is: 10880 (see Functional Point Count capture)

Hence, the ratio DBbyte/SLOC is: 446000/10880 = 41

Product Complexity(CPLX)

[average: 1,05]

The analysis is done on the entire system, considering always the worst possible case.

* Control Operations [HIGH - 1,17]: the system uses queues and complex data structures, in order to handle users, cars, reservation request.

* Computational Operations [LOW - 0,87]: the system evaluates very simple computational operations.

Examples:

1) user profile statistics (additions)

2) find the nearest car within a range (distance from 2 points formula)

3) bonus calculation (percentages calculation)

4) payment calculation (addition, multiplication)

* Device-dependent Operations [NOMINAL 1,00]: I/O processing includes device selection, status checking, and error processing.

* Data Management Operations [HIGH 1,17]: the system stores data in a structural way through a DBMS. Therefore, the system seeks, reads and writes the data. Data can also be filtered, analyzed, manipulated.

* User Interface Management Operations [LOW 0,87]: the graphic user interface is composed of simple input forms, buttons, radio buttons, and text. The most complex form

is the one that displays the map. In this case, however, the majority of the work is performed by Google's API.

Developed for Reusability(RUSE)

[NOMINAL - 1,00]

This project has been designed following general good practices of reusability of the components. Each component has been thought to be self-contained with a specific purpose.

Documentation match to Life-cycle needs(DOCU)

[NOMINAL - 1,00]

The project is well documented. A set of starting documents (RASD, DD, ITPD) analyze each aspect of the project, from the requirements to the software components. Also, the code is documented, making it easy to ready and quick to modify.

Platform factors

Execution Time Constraint(TIME)

[NOMINAL - 1,00]

The system processes mainly request-response actions, without CPU bound processes.

Main Storage Constraint(STOR)

[NOMINAL - 1,00]

This constraint is essentially irrelevant, since there are no constraints on the hardware.

Platform Volatility(PVOL)

[LOW - 0,87]

The platform of the system is composed of: operating systems, application server, framework (e.g. J2EE), IDEs, the cars' electronics, smartphones' hardware. Each of them is receive a major update (both hardware and software) at most 1 time for year.

Personnel Factors

Analyst Capability (ACAP)

[NOMINAL - 1,00]

The analysts of this project are MSc graduated students without experience. Their decisions are driven by guidelines, good practices, and personal knowledge.

Programmer Capability (PCAP)

[NOMINAL - 1,00]

The programmers are the same people that have designed the project. They know very well the principles of objective oriented programming languages and the main patterns.

Personnel Continuity (PCON)

[VERY HIGH - 0,81]

The project has been followed by the team from the beginning to the end.

Applications Experience (APEX)

[LOW - 1,10]

The team has almost no experience in the developing an application of this kind.

Platform Experience (PLEX)

[LOW - 1,09]

The majority of the software and hardware that the developers are supposed to deal with is almost unknown.

Language and Tool Experience (LTEX)

[LOW - 1,09]

This level is an average value considering a good knowledge of Java programming language, a moderate one of J2EE, and a poor one of C# and Objective-C.

Project Factors

Use of Software Tools (TOOL)

[VERY LOW - 1,17]

CASE tools were not used during the development of the project.

Multisite Development (SITE)

[Extra High - 0,80]

The team is very small (2 people) and have worked together, in the same room. They also use instant message clients to talk.

General Factor

Required Development Schedule (SCED)

[NOMINAL - 1,00]

The schedule of the project has been completely respected, without rushing before the milestones.

DATA COLLECTION

SCALE FACTOR

PREC	0,05
FLEX	0,03
RESL	4,24
TEAM	0,01
PMA	0,05

POST-ARCHITECTURE EFFORT ESTIMATION

RELY	1,00
DATA	1,00
CPLX	1,05
RUSE	1,00
DOCU	1,00
TIME	1,00
STOR	1,00
PVOL	0,87
ACAPO	1,00
PCAP	1,00
PCON	0,81
APEX	1,10
PLEX	1,09
LTEX	1,09
TOOL	1,17
SITE	0,80
SCED	1,00

PERSON MONTHS CALCULATION

$$PM = A * SIZE^{E} * \prod_{1 \leq i \leq n} EM_i$$

Constants and results

A	2,94
E	0,9538
B	0,91
KSLOC	10,88
PM	26
PEOPLE	2
TOTAL MONTHS	**13**

Phases estimation

In the following table all the different phases of the project are listed. Each phase is associated with its duration in percentage with respect to the total duration of the project and the corresponding duration in months.

The Analysis and Design phase lasted about 3 months that is approximately 23% of the total time. The development phase takes up about 32% of the total time. The testing phase, divided in unit testing, system testing and acceptance test, takes up about 30% of the project schedule. The deployment phase and the maintenance/technical support phase take up 10% of the total time.

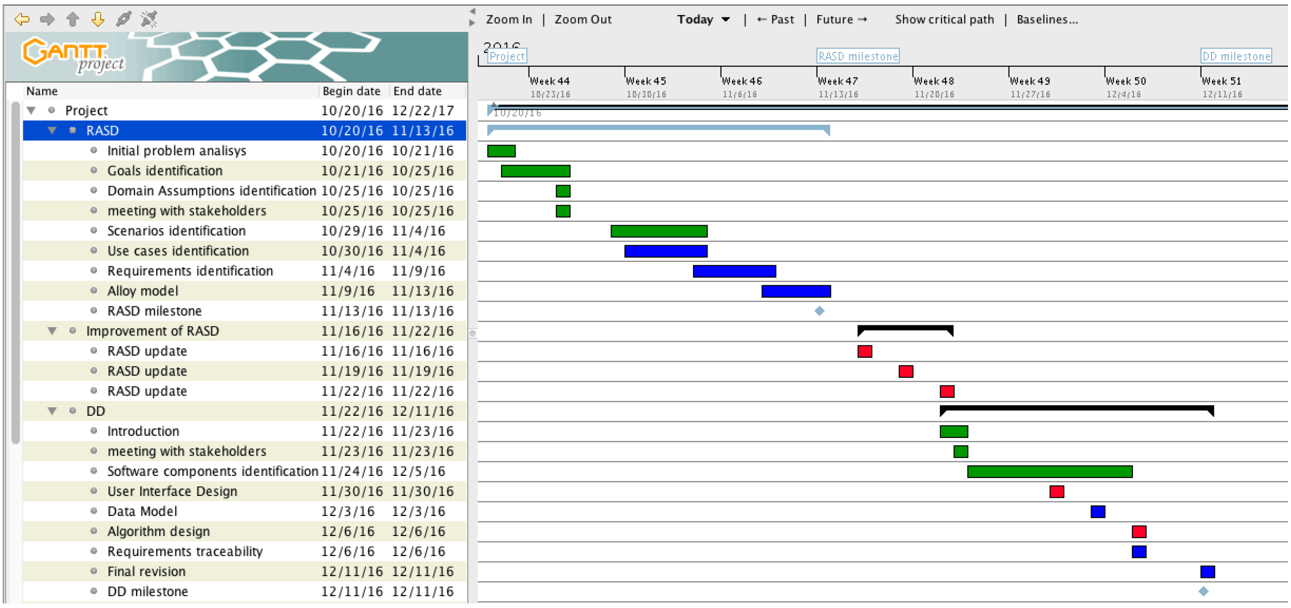
Phase	Percentage	Months
Analysis and Design	23 %	3
Development	32 %	4,16 (+1 extra)
Unit testing	10 %	1,3
System testing	10 %	1,3
Deployment	10 %	1,3
Acceptance test	10 %	1,3
Maintenance/Technical support	10 %	1,3

3. PLANNING PROCESS AND RESOURCE ALLOCATION

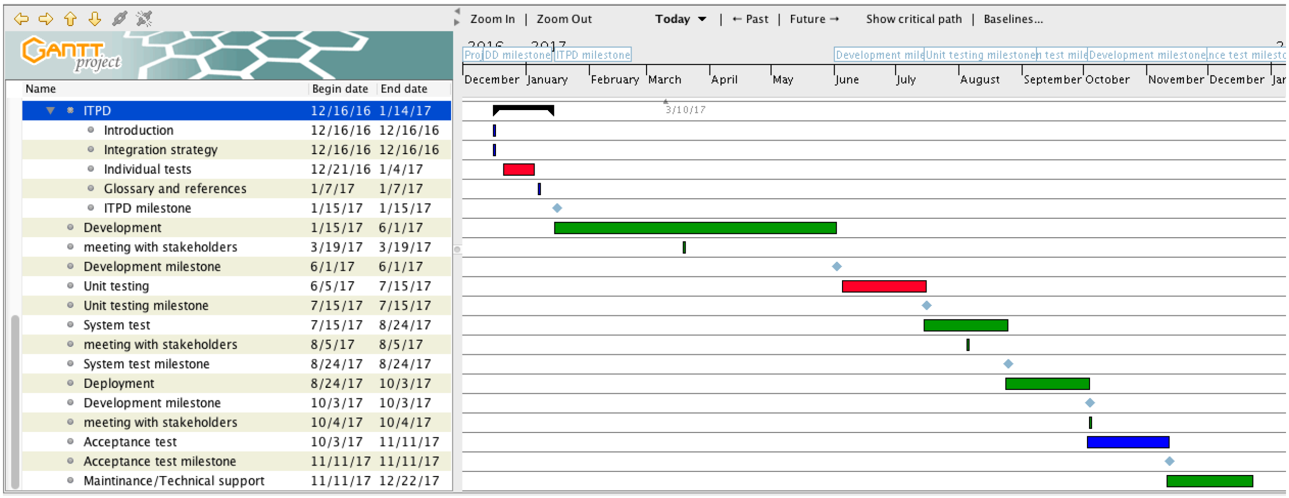
In the following diagrams is shown the expected schedule of the project calculated in the previous paragraph.

The green colored tasks were performed by the whole team.

The blue ones were performed by Francesco Peverelli and the red ones by Federico Reppucci.



Name	Begin date	End date
Project	10/20/16	12/22/17
RASD	10/20/16	11/13/16
Initial problem analisys	10/20/16	10/21/16
Goals identification	10/21/16	10/25/16
Domain Assumptions identification	10/25/16	10/25/16
meeting with stakeholders	10/25/16	10/25/16
Scenarios identification	10/29/16	11/4/16
Use cases identification	10/30/16	11/4/16
Requirements identification	11/4/16	11/9/16
Alloy model	11/9/16	11/13/16
RASD milestone	11/13/16	11/13/16
Improvement of RASD	11/16/16	11/22/16
RASD update	11/16/16	11/16/16
RASD update	11/19/16	11/19/16
RASD update	11/22/16	11/22/16
DD	11/22/16	12/11/16
Introduction	11/22/16	11/23/16
meeting with stakeholders	11/23/16	11/23/16
Software components identification	11/24/16	12/5/16
User Interface Design	11/30/16	11/30/16
Data Model	12/3/16	12/3/16
Algorithm design	12/6/16	12/6/16
Requirements traceability	12/6/16	12/6/16
Final revision	12/11/16	12/11/16
DD milestone	12/11/16	12/11/16


Zoom In | Zoom Out
Today
← Past | Future →
Show critical path | Baselines...
2016 2017
Proj DD milestone ITPD milestone
Development mil Unit testing milestone test mile Development milestone nce test milest
December January February March April May June July August September October November December Jar
3/10/17

Name	Begin date	End date
ITPD	12/16/16	1/14/17
Introduction	12/16/16	12/16/16
Integration strategy	12/16/16	12/16/16
Individual tests	12/21/16	1/4/17
Glossary and references	1/7/17	1/7/17
ITPD milestone	1/15/17	1/15/17
Development	1/15/17	6/1/17
meeting with stakeholders	3/19/17	3/19/17
Development milestone	6/1/17	6/1/17
Unit testing	6/5/17	7/15/17
Unit testing milestone	7/15/17	7/15/17
System test	7/15/17	8/24/17
meeting with stakeholders	8/5/17	8/5/17
System test milestone	8/24/17	8/24/17
Deployment	8/24/17	10/3/17
Development milestone	10/3/17	10/3/17
meeting with stakeholders	10/4/17	10/4/17
Acceptance test	10/3/17	11/11/17
Acceptance test milestone	11/11/17	11/11/17
Maintinance/Technical support	11/11/17	12/22/17

4. RISK MANAGEMENT

4.1 RISK MANAGEMENT PLAN INTRODUCTION

In the following paragraph the main risk associated with the project are presented and evaluated in terms of potential loss and damage. Furthermore, different strategies which have been devised in order to reduce the risk and damage are explained.

All the risk presented are identified as either Low, Medium or High from a probability standpoint, and categorized as negligible, marginal, critical or catastrophic with respect to the impact anticipated.

4.2 RISK FACTORS ANALYSIS

4.2.1 PROJECT RISKS

1. Staff Illness: this risk factor takes into account the eventuality in which one or more member of the team are incapacitated due to illness or injury. Although the probability is considered Low, due to the small size of the development team the impact of such risk is considered critical. The countermeasure devised to overcome the issue consists in both members being involved in virtually all the phases of the project and in making an effort to communicate effectively, so that the other member of the team may advance with the project alone for some time if need be.

2. Optimistic Schedule: this risk factor represents the possibility for the project schedule devised in the Project Plan to be overly optimistic. The probability of this occurrence is Medium, as the experience of the team in project planning is rather low, but the size of the project is not overwhelming. The entity of the anticipated impact is marginal, since there are no deadlines which threat the feasibility of the project, from the stakeholders' perspective. The strategy adopted to overcome this inconvenience consists of allocating some extra days in the project schedule, and reserving some the budget to look for "off the shelf" components which may speed-up the development process.

3. Gold plating: this risk factor represents the possibility of an unrealistic and ever-changing quality standard hindering the development process. The probability of this risk is considered Low, since the team is small and shares the same view on the matter, placing the timely delivery of the product among the top priorities. The associated impact is also marginal, for the same reasons stated for the Optimistic Schedule risk factor, since the resulting effect would ultimately be a delay in the project schedule. The countermeasure devised to avoid this risk is a prioritization of the development of all core functionalities before any improvement on any existing functionality or feature.

4. Wrong functionality: this risk factor accounts for the possibility for a functionality of the software product to be either useless or different from what the stakeholder wanted. The

probability for this risk is considered low, since an accurate analysis of the requirements as been carried out, also involving the stakeholders in the process. Nevertheless, if this risk were to become a reality, the impact is to be considered Critical, since it may heavily affect either the quality of the resulting software product or its timely release. The strategy devised to prevent this risk involves actively meet the stakeholders to verify the fulfillment of the requirements at different stages of the development process.

5. Team Skills: this risk factor represents the possible lack of the necessary skills needed to develop the software from the development team. The probability of this event is Low, as the skills of the development team are well-known, considering the size of the team itself, and such skills are considered to be sufficient to deliver the software product. On the other hand, if the lack of skills from the developers were to manifest itself it would be difficult to face this issue, therefore the estimated impact is Critical. In order to overcome the issue, the estimation for the schedule takes into account the possibility that the team may need to acquire the necessary skills as the project is being developed.

4.2.2 BUISNESS RISKS

1. Low Budget Estimate: this risk factor accounts for the possibility of an error in the estimated budget for the project. Since the costs for the cars and the power grid stations are known in advance, as well as the cost of the hardware to buy, the probability of this occurrence is considered Low. Depending on the entity of the error however, the corresponding impact can be considerable, therefore the risk is categorized as Marginal in this respect. The preventive measures taken are a worst-case-scenario estimate of the budget, and the presence of a margin on top of the estimate, used to account for a potential error.

2. Requirements Volatility: this risk factor represents the possibility for the requirement of the software product to change over time, either for a change in the environment in which it operates or a change in the need of the stakeholders. The probability of this occurrence is considered High, since the requirements for a software product of this kind are expected to change often, either for the presence of competitors or changes in the external software components which interact with the system. On the other hand, the impact of this risk is considered Low, as it is a rather expected event which does not impact the availability of the service in most cases. The countermeasures in place to reduce the impact of this risk include a flexible software architecture and a high degree of decoupling among component, which makes the removal or addition of functionalities easier and less costly.

3. Organizational Financial Problems: this risk factor accounts for possible difficulties in obtaining the pre-planned budget due to financial problems within the organization. This risk factor's probability is considered Medium, as the organization seems committed in providing the adequate founding, but the overall cost of the project is not negligible. With

respect to the associated risk this occurrence is considered Catastrophic, as it may very well mean the cancellation of the project altogether.

4. Market Risk: this risk factor represents the eventuality in which the PowerEnJoy service does not find the expected success on the market. This possibility is estimated to be Low, since the appropriate research has been conducted by the stakeholders in advance. On the other and, the impact in case this risk was to become a reality is considered Critical, as it would mean a failure in the very purpose of the project, which is to provide a useful and meaningful service to the customer.

5. Legislation Changes: this risk factor represents the possibility for legislation changes to limit or prevent the use of the service provided by the project. Considering that a car sharing service is overall beneficial to a city, especially considering that the use of electric cars further reduces the overall emissions being produced in said city, there is no foreseeable reason for which specific legislation should be issued in the direction of impairing or banning the service provided by the project, therefore the estimated probability is Low. Depending on the content of the legislation in question however the effect may very well be considered Critical. The main preventive measure which has been taken is an extensive research from the company requesting the software product in order to make sure that the service does not go against any law or regulation.

4.2.3 TECHNICAL RISKS

1. Defective / Lacking External Components: this risk factor takes into account the possibility that one or more external components do not provide the expected functionality either with sufficient quality or at all. The probability of this occurrence is considered low, since the services chosen are already widely adopted and successful. The corresponding impact is considered marginal, since although these services can be easily substituted with similar ones from other providers, this change may result in some work on the corresponding interfaces of the software product. The strategy devised to avoid such risk consists of an accurate research to select the best possible services in the respective areas, and gain a clear understanding of the services offered, in order to make sure that the service selected fulfills the role correctly. In addition, during the development of the interfaces a lot of priority has to be placed into making sure that they are general enough to allow a smooth transition to another similar service.

2. Real Time Performance Shortfalls: this risk factor represents the possibility of the software suffering from performance shortfalls. The probability of this event is considered Low, since the expected number of concurrent user has been estimated and the architecture selected, both from a hardware and a software perspective, is robust in this respect. The estimated impact of this event is Marginal, as it may represent an issue which discourages the users from making use of the service, but is not seen as a critical issue, as

it is relatively easily addressable. In fact, in case of performance shortfalls additional hardware may be bought to relieve some of the burden on the existing machines.

3. Technology Issues: this risk factor takes into account the possibility that the changes in technology standards may represent an issue for the software product. The probability of this occurrence is considered Low, since the main development framework is very stable, and even though the standards and technologies for the different smartphone operative systems can be subject to rapid changes, they are often retro compatible. The impact of this risk is marginal, since in most cases a few adjustments should solve the issue at hand.

4. Loss of Source Code: this risk factor accounts for the possibility that the source code of the project may be lost, all or in part. The possibility of this occurrence is Low, as proper version control software is being used, as well as relatively frequent backups. Although improbable, the estimated impact of this risk is Catastrophic, as it may result in either a massive delay in the project schedule, or the cancellation of the project.

5. USED TOOLS

GanttProject - www.ganttproject.biz

6. EFFORT SPENT

Together: [6h]

Reppucci: [8h]

Peverelli: [6h]