Politecnico di Milano

Software Engineering II project

PowerEnJoy

Project Plan

Authors:
Davide Piantella
Mario Scrocca
Moreno R. Vendra

Professor:
Luca Mottola

January 22, 2017

version 1.0

Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose of this document

The purpose of the *Power EnJoy Project Plan* is to provide an overview of the project with respect to parameters related to the project management. This document provides an estimation of the expected size, cost and effort for the project based upon techniques that support the estimation procedure (Function Points, COCOMO II); it also proposes a possible schedule for the project and a related allocation of resources to accomplish tasks identified and describes the possible risks related to the project presenting some possible strategies to manage them.

1.2 Scope

PowerEnJoy is a car-sharing service that exclusively employs electric cars; we are going to develop a web-based software system that will provide the functionalities normally provided by car-sharing services, such as allowing the user to register to the system in order to access it, showing the cars available near a given location and allowing a user to reserve a car before picking it up. A screen located inside the car will show in real time the ride amount of money to the user. When the user reaches a predefined safe area and exits the car, the system will stop charging the user and will lock the car. The system will provide information about charging station location where the car can be plugged after the ride and incentivize virtuous behaviours of the users with discounts.

1.3 Glossary

1.3.1 Acronyms

RASD: Requirements Analysis and Specification Document

DD: Design Document

ITPD: Integration Test Plan Document

FP: Function Points

ILF: Internal logic file

EIF: External interface file

EI: External Input

EO: External Output

EQ: External Inquiries

DET: Data Element Type

RET: Record Element Type

FTR: File Type Referenced

COCOMO: COnstructive COst MOdel

SLOC: Source Line Of Code

API: Application Programming Interface

GPS: Global Position System

DB: DataBase

DBMS: DataBase Management System

GIS: Geographic Information System

1.3.2 Abbreviations

w.r.t.: with respect to

i.d.: id est

e.g.: exempli gratia

etc.: et cetera

1.4 Reference documents

- Assignments Software Engineering 2,
  E. Di Nitto, L. Mottola, AA 2016-2017
- Function Point Counting Practices Manual,
  Release 4.2 International Function Point Users Group (IFPUG)
- COCOMO II.2000 – Model Definition Manual,
  Center for Software Engineering, USC

1.5 Document overview

This document is structured as

1. **Introduction**: contains references, glossary, definitions, acronyms and abbreviations; it also explains the purpose and scope of this document
2. **Project size, cost and effort estimation**: provides estimations of the expected size, cost and required effort of the *PowerEnJoy system*
   - Size estimation: *Function Points* will be used to estimate the size of the *PowerEnJoy system* starting from the complexity of its main functionalities
   - Cost and effort estimation: the *Constructive Cost Model (COCOMO) II* will be used to estimate the cost and effort needed to develop the *PowerEnJoy system*
3. **Schedule**: contains a general high level schedule of the project's activities
4. **Resource allocation**: defines how the activities in the schedule will be allocated between the members of the group
5. **Risk management**: analyses the main risks that the project development may face and proposes the strategies to manage

2 Project size, cost and effort estimation

This section provides an estimation for the size, cost and effort of the project based upon techniques that support the estimation procedure.

The *Function Points* method is used to estimate the size of the project while the *COCOMO II* method is used for cost and effort estimations..

2.1 Size estimation: Function Points

Given the assumption that the dimension of software can be characterized based on the functionalities that it offers, Function Points (FP) analysis is a process used to estimate software functional size. The most pervasive version of FP analysis is regulated by the International Function Point User Group (IFPUG).

Functional components can be clustered as: Internal Logical Files (ILF), External Interface Files (EIF), External Inputs (EI), External Outputs (EO) and External Inquiries (EQ).

Each functional component is associated with a *complexity level* based on its associated file and data structure and cardinality. Measures and dimensions are: Data Element Types (DET), File Types Referenced (FTR) and Record Element Types (RET).

Each complexity level is then converted into a *weight coefficient* which is used in the FP formulae to compute overall estimations.

Complexity levels and their weight Table 1, Table 2 and Table 3 describe how complexity level is mapped on each function component.

In Table 4 is shown how each function component is then assigned a weight according to its complexity.

	DET		
RET	1-19	20-50	51+
1	Low	Low	Avg
2-5	Low	Avg	High
6+	Avg	High	High

Table 1: Complexity for ILF and EIF

	DET		
FTR	1-4	5-15	16+
0-1	Low	Low	Avg
2	Low	Avg	High
3+	Avg	High	High

Table 2: Complexity for EI

	DET		
FTR	1-5	6-19	20+
0-1	Low	Low	Avg
2-3	Low	Avg	High
4+	Avg	High	High

Table 3: Complexity for EO and EQ

Component	Low	Average	High
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6
Internal Logic Files	7	10	15
External Interface Files	5	7	10

Table 4: Function component complexity weight assignment

Computing FP The Function Point (FP) is computed as

$$FP = \sum_{i \in I} \sum_{j \in i} w_{ij}$$

where $I = \{ILF, EIF, EI, EO, EQ\}$ and w_{ij} is the weight associated with the j-th function component of type i.

Computing SLOC To be able to convert FP into Source Lines Of Code (SLOC) a language-dependent factor must be taken into consideration. Since J2EE was chosen as coding language for our system, we report in Table 5 the industry gearing distribution of J2EE SLOC/FP factor. [4]

Average	**Median**	**Low**	**High**
46	49	15	67

Table 5: J2EE SLOC/FP factor distribution

Notes The *Function Points* analysis described does not take into account aspects regarding the users interface.

2.1.1 Internal Logic Files (ILFs)

The *Function Points* method explains how to identify ILFs function points through the table provided defining ILF, DET and RET [5]:

- An *internal logical file* (ILF) is a user identifiable group of logically related data or control information maintained within the boundary of the application; the primary intent of an ILF is to hold data maintained through one or more elementary processes of the application being counted
- A *data element type* (DET) is a unique user recognizable, non-repeated field
- A record element type (RET) is a user recognizable subgroup of data elements within an ILF or EIF

Notes Given that definitions, ILFs function points must be calculated independently from the technology chosen for the representation of data, omitting fields used by the system to manage data but non user recognizable (e.g. primary key generated to distinguish tuples in a relational database but without a meaning outside the actual representation).

This section describes the ILFs and related RETs and DETs identified for the *PowerEnJoy system*.

User The system need to store information about user's personal info (name and surname, date of birth and place of birth, address), user's login info (username and password), user's payment info (credit card number), user's driving license (driving license number), user's status (status and banned reason).

Rent payment The system needs to store information about three subgroups of data related to rent payments:

- **Payments**: for each payment the system needs to store data about the rent and the user related to the payment, payment status, payment timestamp, base cost of the ride, applied discount and applied fee
- **Fees**: for each fee applicable to a rent the system needs to store data about fixed rate and percentage rate of the fee
- **Discount**: for each discount applicable to a rent the system needs to store data about fixed rate and percentage rate of the fee

Rent For each rent the system needs to store data about start timestamp and start location, end timestamp and end location, payment related to the rent, user performing the rent, car rented and eventually charging station related to the money saving option.

Reservation For each active reservation the system needs to store data about start timestamp, user who made the reservation, car reserved and eventually charging station related to the money saving option.

Car The system needs to store information about two subgroups of data related to cars:

- **Car data**: for each car the system needs to store data about the car status, the GPS position, the battery level, the license number, the model and a last-seen-timestamp to know last time information about other fields has been updated
- **Failures**: for each failure (fixed or not) the system needs to store data about problem description, report date and eventually the date when the failure has been fixed

Notes Some data stored by the system are retrieved from cars trough the provided API primitives, these data represent a cache copy of the actual data of the car which instead are considered as an EIF.

Charging Station For each charging station the system needs to store data about GPS location and number of plugs.

Safe Area The system needs to store information about two subgroups of data related to safe areas:

- **Closed polygonal chain**: for each safe area the system needs to store data about points composing the safe area and their order in the closed polygonal chain composing it
- **Point of the polygonal chain**: for each point the system needs to store the GPS coordinates

ILF	RET	DET	Complexity	**FPs**
User	1	10	Low	7
Rent payment	3	11	Low	7
Rent	1	8	Low	7
Reservation	1	4	Low	7
Car	2	9	Low	7
Charging Station	1	3	Low	7
Safe Area	2	3	Low	7
Total				**49**

Table 6: Function points ILFs

2.1.2 External Interface Files (EIFs)

The *Function Points* method explains how to identify EIFs function points through the table provided defining EIF, DET and RET.

An *external interface file* (EIF) is a user identifiable group of logically related data or control information referenced by the application, but maintained within the boundary of another application. The primary intent of an EIF is to hold data referenced through one or more elementary processes within the boundary of the application counted. This means an EIF counted for an application must be in an ILF in another application [5].

DET and RET are defined as described in the ILFs section.

This section describes the EIFs and related RETs and DETs identified for the *PowerEnJoy system*.

Location The system relies on data about mapping of location and GPS position of the GIS API.

Car The system relies on data provided by the car API: unique car identifier, GPS position, engine status (on, off), current number of passengers, maximum number of passengers, battery level (percentage), charging status (in charge, not in charge), door locking status (locked, unlocked), door closing status (open, closed).

Notes The system relies on data about maps provided by the GIS API but only needs a map reference to allow the user client application to show a map centred in a given position, with a given zoom and a set of pointer and areas (data managed by the system and already considered as ILF), therefore map data are not considered as an EIF.

EIF	RET	DET	Complexity	**FPs**
Location	1	2	Low	7
Car	1	9	Low	7
Total				**14**

Table 7: Function points EIFs

2.1.3 External Inputs (EIs)

The *Function Points* method explains how to identify EIs function points through the table provided defining EI, FTR and DET [5]:

- An *external input* (EI) is an elementary process that processes data or control information that comes from outside the application boundary. The primary intent of an EI is to maintain one or more ILFs and/or to alter the behavior of the system.
- *A file type referenced* (FTR) is:
  - An internal logical file read or maintained by a transactional function *or*
  - An external interface file read by a transactional function
- A *data element type* (DET) is a unique user recognizable, non-repeated field

This section describes the EIs and related FTRs and DETs identified for the *PowerEnJoy system* grouped by the user category.

Guest user

- **Register** to the *PowerEnjoy system*: to be processed by the system only requires users data to instantiate a new user record

Registered user

- **Authenticate** to the *PowerEnjoy system*: to be processed by the system only requires users data to check inserted credentials
- **Edit user info**: to be processed by the system only requires users data to modify information stored by the system as requested
- **See available cars**: to be processed by the system it requires cars info (data stored in the system if recently updated, or data retrieved from cars otherwise), charging stations and safe areas data to show it on the map, and eventually data provided by GIS API to decode the position inserted by the user
- **Reserve a car** (we consider the most complete scenario with money saving option enabled): to be processed by the system it requires cars info, access to active reservations data to instantiate a reservation record, charging stations data to provide the recommended destination and eventually data provided by GIS API to decode the position inserted by the user
- **Reservation expired**: to be processed by the system it requires active reservations info to delete the reservation and payments data to instantiate a new payment record
- **Unlock a car**: to be processed by the system it requires cars info and active reservations data to check if the request is allowed for the user making it

- **Start a rent** (the system is notified that the user has ignited the car engine): to be processed by the system it requires rents data to create a new record rent (inserting start timestamp and location), car data to change car status and active reservations info to delete the reservation record
- **End a rent** (the system is notified that the user has turned engine off, has leaved the car and has closed doors): to be processed by the system it requires to retrieve data from cars, requires to modify rents data and access to payments data to instantiate a new payment record and calculate the applicable fees and discount and the final total cost

Customer care user

- **Ban or enable a user**: to be processed by the system it only requires users data to change user status
- **Tag a car as *Not available***: to be processed by the system it only requires cars data to change car status, instantiate a new failures record and its fields

Maintenance API

- **Retrieve *Not Available* cars**: to be processed by the system it requires cars and failures data but also to retrieve from cars software keys to access them (so it can't be considered as a an EQ)
- **Tag a car as *Available***: to be processed by the system it only requires cars data to change car status and mark the failure as fixed

Car Event

- **Critical battery level**: to be processed by the system it only requires cars data to change car status as *Not available*, instantiate a new failures record and its fields

EI	FTR	DET	Complexity	**FPs**
Register	1	5-15	Low	3
Edit user info	1	1-4	Low	3
Authenticate	1	1-4	Low	3
See available cars	3+	5-15	High	6
Reserve a car	3+	5-15	High	6
Reservation expired	2	5-15	Avg	4
Unlock a car	2	1-4	Low	3
Start a rent	3+	5-15	High	6
End a rent	3+	5-15	High	6
Ban or enable a user	1	1-4	Low	3
Tag a car as *Not available*	1	1-4	Low	3
Retrieve *Not available* cars	2	5-15	Avg	4
Tag a car as *Available*	1	1-4	Low	3
Critical battery level	1	1-4	Low	3
Total				**56**

Table 8: Function points EIs

2.1.4 External Outputs (EOs)

The *Function Points* method explains how to identify EOs function points through the table provided defining EO, FTR and DET.

An *external output* (EO) is an elementary process that sends data or control information outside the application boundary. The primary intent of an external output is to 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 [5].

DET and FTR are defined as described in the EIs section.

This section describes the EOs and related FTRs and DETs identified for the *PowerEnJoy system* grouped by the user category.

Payment API

- **Make a payment procedure**: to be executed by the system through the payment API it requires data about the payment to be made and information about user's payment method. It also requires to modify user status if the payment procedure fails

GIS API

- **Decode an address as GPS position**: to be executed by the system through the GIS API it requires the address to be decoded and access to location EIF related to GIS API

- **Obtain a map reference**: to be executed by the system and to retrieve the reference through the GIS API it requires the GPS position of the user to center the map, the zoom and information about cars (data stored in the system if recently updated, or data retrieved from cars otherwise), safe areas and charging stations

Cars

- **Car primitive**: to be executed by the system through the cars provided API it requires access to data about cars to identify car to communicate with and eventually to update fields related. Number of DET related may depend on the primitive called so we consider that EO of *Average* difficulty

EO	FTR	DET	Complexity	**FPs**
Make a payment procedure	2-3	6-19	Avg	5
Decode an address as GPS position	1	1-5	Low	4
Obtain a map reference	4+	6-19	Avg	7
Car primitive	1	-	Avg	5
Total				**21**

Table 9: Function points EOs

2.1.5 External Inquiries (EQs)

The *Function Points* method explains how to identify EQs function points through the table provided defining EQ, FTR and DET.

An *external inquiry* (EQ) is an elementary process that sends data or control information outside the application boundary. The primary intent of an external inquiry is to present information to a user through the retrieval of data or control information from an ILF of EIF. The processing logic contains no mathematical formulas or calculations, and creates no derived data. No ILF is maintained during the processing, nor is the behavior of the system altered [5].

DET and FTR are defined as described in the EIs section.

This section describes the EQs and related FTRs and DETs identified for the *PowerEnJoy system* grouped by the user category.

Registered user

- **See personal information**: to be processed by the system it requires the userID and data about users
- **See rent history**: to be processed by the system it requires the userID and data about rents
- **See payment history**: to be processed by the system it requires the userID and data about payments

Customer care user

- **See all user IDs**: to be processed by the system it requires data about users
- **See information about a user**: to be processed by the system it requires the userID and data about users
- **See rent history of a user**: to be processed by the system it requires the userID and data about rents
- **See payment history of a user**: to be processed by the system it requires the userID and data about payments

Notes Similiar EQs between users and customer care operator are considered separately because are requested from different server software components and they also differ in information retrieved as EQ result.

EQ	FTR	DET	Complexity	**FPs**
See own information (*user*)	1	6-19	Low	4
See own rent history (*user*)	1	6-19	Low	4
See own payment history (*user*)	1	6-19	Low	4
See all user IDs	1	1-5	Low	4
See information about a user	1	6-19	Low	4
See rent history of a user	1	6-19	Low	4
See payment history of a user	1	6-19	Low	4
Total				**28**

Table 10: Function points EQs

2.1.6 Overall estimation

The following table summarizes FPs count identified through the analysis for each component.

Function Type Value	FPs
Internal Logic Files	49
External Interface Files	14
External Inputs	56
External Outputs	21
External Inquiries	28
Total	**168**

Table 11: Function points

Through the $SLOC$ estimation formula we can therefore estimate an average and an upper bound for the size of the project.

$$SLOC = (\sum_{i \in I} \sum_{j \in i} w_{ij}) \cdot K$$

where

- I = {ILF,EIF,EI,EO,EQ}
- w_{ij} is the weight associated with the j-th function component of type i
- K is the industry gearing distribution of J2EE $\frac{SLOC}{FP}$ factor

$$SLOC_{avg} = (\sum_{i \in I} \sum_{j \in i} w_{ij}) \cdot K = 168 \cdot 46 = 7728$$
$$SLOC_{max} = (\sum_{i \in I} \sum_{j \in i} w_{ij}) \cdot K = 168 \cdot 67 = 11256$$

2.2 Cost and effort estimation: COCOMO II

COnstructive COst MOdel II (COCOMO II) is a model that allows one to estimate the cost, effort, and schedule when planning a new software development activity. COCOMO II.2000 is the latest major extension to the original COCOMO 81 model published in 1981.

It consists of three submodels, each one offering increased fidelity the further along one is in the project planning and design process [6].

Since the system we are to develop is brand new and there is no previous system we have to adopt the *early design* submodel.

Important note All parameters and coefficients used in this section are formally defined in the *COCOMO II.2000 Model Definition Manual* [7]. Each cost driver and scale factor value is determined w.r.t. definition tables in the same manual that are not reported here.

COCOMO effort equation This formula estimates the project effort in Person-Months (PM)

$$PM = A \cdot Size^{E} \cdot \prod_{i \in CostDrivers} C_i$$

where

- $A = 2.94$ approximates a productivity constant in $\frac{PM}{KiloSLOC}$
- $Size$ is the estimated size of the project in KiloSLOC, it can be deducted from Function Points analysis
- E is an aggregation of five Scale Factors which is computed as

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

- C is the effort multiplier derived from each Cost Driver

2.2.1 Scale Factors

Here are described the five Scale Factors of the COCOMO II model, in Table 12 are specified their numerical value according to the model definition.

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprecedented **6.20**	largely unprecedented **4.96**	somewhat unprecedented **3.72**	generally familiar **2.48**	largely familiar **1.24**	thoroughly familiar **0.00**
FLEX	rigorous **5.07**	occasional relaxation **4.05**	some relaxation **3.04**	general conformity **2.03**	some conformity **1.01**	general goals **0.00**
RESL	little (20%) **7.07**	some (40%) **5.65**	often (60%) **4.24**	generally (75%) **2.83**	mostly (90%) **1.14**	full (100%) **0.00**
TEAM	very difficult interactions **5.48**	some difficult interactions **4.38**	basically cooperative interactions **3.29**	largely cooperative **2.19**	highly cooperative **1.10**	seamless interactions **0.00**
PMAT	SW-CMM Level 1 Lower **7.80**	SW-CMM Level 1 Upper **6.24**	SW-CMM Level 2 **4.68**	SW-CMM Level 3 **3.12**	SW-CMM Level 4 **1.56**	SW-CMM Level 5 **0.00**

Table 12: COCOMO II scale factors values

Precedentedness - PREC It is high if a product is similar to several projects previously developed by the team. Since we have never develop such a big project, although we have largely understood the product objectives and there is a minimal need of innovative algorithms or innovative data processing architectures, we decide that

PREC is **LOW**

Development Flexibility - FLEX It is high if there are no specific constraints to conform to pre-established requirements and external interface specifications. The italian law establishes specific constraints regarding the characteristics and usage of car sharing systems, driving licenses and privacy policy. Requirements specified by the client do not excessively restrict the development process, some external APIs should be used in order to fulfill the requirements. We decide that

FLEX is **NOMINAL**

Risk Resolution - RESL It is high if the project has a good risk management plan, clear definition of budget and schedule, focus on architectural definition. The Risk Management Plan identifies generally all critical risk items and determines actions in order to resolve them. As specified in the Schedule section a relevant portion of development process is devoted to establishing architecture, given general product objectives.
Therefore we decide that

RESL is **HIGH**

Team Cohesion - TEAM It is high if all project development team members are able to work in a team and share the same vision and commitment. As objectives, cultures, ages and backgrounds are the same for all team members we decide that

TEAM is **VERY HIGH**

Project Maturity - PMAT It reflects the CMMI index of the project. Since this project can be considered a managed process, planned and executed according to policies with the usage of adequate and planned resources, stakeholders constantly involved in incremental product reviews, we decide that the CMMI index is Level 2: Managed at the Project Level, therefore

PMAT is **NOMINAL**

E parameter In the Table 13 are reported the values chosen for the five Scale Factors

Scale Factor	Value	Numerical Value
PREC	Low	4.96
FLEX	Nominal	3.04
RESL	High	2.83
TEAM	Very High	1.10
PMAT	Nominal	4.68
Total		**16.61**

Table 13: Scale factors chosen values

The E parameter of COCOMO effort equation is computed as

$$E = 0.91 + 0.01 \cdot \sum_{j=1}^{5} SF_j = 0.91 + 0.01 \cdot 16.61 = 1.0761$$

2.2.2 Cost Drivers

Since we refer to the *Early Design* model of COCOMO II, the cost drivers are obtained averaging their *Post-Architecture* counterparts as shown in Table 14.

Early Design Cost Drivers	Combined Post-Architecture Cost Drivers
PERS	ACAP, PCAP, PCON
RCPX	RELY, DATA, CPLX, DOCU
RUSE	RUSE
PDIF	TIME ,STOR, PVOL
PREX	APEX, PLEX, LTEX
FCIL	TOOL, SITE
SCED	SCED

Table 14: Cost drivers conversion

Cost Driver	Extra Low	Very Low	Low	Nominal	High	Very High	Extra High
PERS	2.12	1.62	1.26	1.00	0.83	0.63	0.50
RCPX	0.49	0.60	0.83	1.00	1.33	1.91	2.72
RUSE	-	-	0.95	1.00	1.07	1.15	1.24
PDIF	-	-	0.87	1.00	1.29	1.81	2.61
PREX	1.59	1.33	1.22	1.00	0.87	0.74	0.62
FCIL	1.43	1.30	1.10	1.00	0.87	0.73	0.62
SCED	-	1.43	1.14	1.00	1.00	1.00	-

Table 15: COCOMO II early design cost driver values

Personnel Capability - PERS

- **Analyst Capability - ACAP**: Analysts are personnel who work on requirements, high-level design and detailed design. Since we consider ourselves as good analysts, with the ability to communicate and cooperate, we consider this parameter as **HIGH**.
- **Programmer Capability - PCAP**: This parameter reflect the capability of the programmers as a team rather than as individuals. Since all team members have already done with profit several team projects before, we consider this parameter as **HIGH**.
- **Personnel Continuity - PCON**: The rating scale is in terms of the project's annual personnel turnover. Since the team is composed of 3 members and the average turnover is 2 years, this parameter is **VERY LOW**.

PERS		
Cost Driver	Value	Numerical Value
ACAP	High	4
PCAP	High	4
PCON	Very Low	1
Total		**9**

Table 16: PERS aggregated value

Therefore PERS is **NOMINAL**.

Product Reliability and Complexity - RCPX

- **Required Software Reliability - RELY** This is the measure of the extent to which the software must perform its intended function over a period of time. If the effect of a software failure is only slight inconvenience then RELY is very low. Since a product failure will cause high financial loss to our client's core business, this parameter is **HIGH**.

- **Data Base Size - DATA** This cost driver attempts to capture the effect large test data requirements have on product development. The rating is determined by calculating D/P, the ratio of bytes in the testing database to SLOC in the program. Given the project specifications we estimate a testing database size of 10Mb and the FP analysis estimates SLOC between 7728 and 11256, the D/P ratio is between 932 and 1357. Therefore this parameter is **HIGH**.

- **Product Complexity - CPLX** Complexity is divided into five areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations.

  - User interface management: simple use of widget set. It is NOMINAL.
  - Data management operations: large operational database with many updates, possibly with triggers, multi-file structured inputs, data restructuring. It is HIGH.
  - Device dependent Operations: I/O processing includes device selection, status checking and error processing. It is NOMINAL.
  - Computational operations: Use of standard math, basic matrix and vector operations. It is NOMINAL.
  - Control operations: Reentrant and recursive coding. Task synchronization, complex callbacks. It is VERY HIGH.

  Due to this analysis, the CPLX parameter is **NOMINAL**.

- **Documentation Match to Life Cycle Needs - DOCU** This cost driver is evaluated in terms of the suitability of the project's documentation. Since the documentation of the project is supposed to be right-sized to life cycle needs, this parameter is **NOMINAL**

RCPX

Cost Driver	Value	Numerical Value
RELY	High	4
DATA	High	4
CPLX	Nominal	3
DOCU	Nominal	3
Total		**14**

Table 17: RCPX aggregated value

Therefore RCPX is **HIGH**.

Developed for Reusability - RUSE This cost driver accounts for the additional effort needed to construct components intended for reuse on current or future projects. This effort is consumed with creating more generic design of software, more elaborate documentation, and more extensive testing to ensure components are ready for use in other applications.

Since some project components must cover general aspects of different possible software projects, this parameter is **NOMINAL**.

Platform Difficulty - PDIF

- **Execution Time Constraint - TIME** This is a measure of the execution time constraint imposed upon a software system. The rating is expressed in terms of the percentage of available execution time expected to be used by the system or subsystem consuming the execution time resource. Since the system may have exceptional spikes of usage, the idle resource allocation should not exceed 70% of available execution time. Therefore this parameter is **HIGH**.
- **Main Storage Constraint - STOR** This rating represents the degree of main storage constraint imposed on a software system or subsystem. Since the system will not require an incremental disk space except for the database whose size is trivial w.r.t modern disk capacity, this parameter is **NOMINAL**.
- **Platform Volatility - PVOL** *Platform* is used here to mean the complex of hardware and software the product calls on to perform its tasks. Our system may use a web server, a DBMS and some external APIs, we expect major releases of them every 6 months and minor releases every 2 weeks, therefore this parameter is **NOMINAL**.

PDIF		
Cost Driver	Value	Numerical Value
TIME	High	4
STOR	Nominal	3
PVOL	Nominal	3
Total		**10**

Table 18: PDIF aggregated value

Therefore PDIF is **NOMINAL**.

Personnel Experience - PREX

- **Applications Experience - APEX** The rating for this cost driver is dependent on the level of applications experience of the project team developing the software system. Since all members of the team have an equivalent level of experience with this type of application which is approximately 4 months, this parameter is **VERY LOW**.

- **Platform Experience - PLEX** This parameter reflect the ability to understand and use powerful platforms, including more graphic user interface, database, networking and so on. Since all members of the team have already deal (in different teams) with projects that relied on platforms similar to the ones that will be used for the PowerEnJoy project for an average period of 2 years, this parameter is set to **NOMINAL**.
- **Language and Tool Experience - LTEX** This is a measure of the level of programming language and software tool experience of the project team developing the software system or subsystem. All team members have quite good knowledge of many development best practices and software tools that will be used such as version control systems, integrated development environments, testing frameworks, development quality control software, etc. None of team members have any experience regarding J2EE. All these things considered, this parameter is **NOMINAL**.

PREX		
Cost Driver	Value	Numerical Value
APEX	Very Low	1
PLEX	Nominal	3
LTEX	Nominal	3
Total		**7**

Table 19: PREX aggregated value

Therefore PREX is **LOW**.

Facilities - FCIL

- **Use of Software Tool - TOOL** Software development tools will be used during the whole development cycle (version control systems, integrated development environments, testing frameworks, development quality control software, bug report systems, virtual team desks, etc.), therefore this parameter is **NOMINAL**.
- **Multisite Development - SITE** Since all team members and stakeholders live in the same metropolitan area, this parameter is **HIGH**.

FCIL		
Cost Driver	Value	Numerical Value
TOOL	Nominal	3
SITE	High	4
Total		**7**

Table 20: FCIL aggregated value

Therefore FCIL is **High**.

Required Development Schedule - SCED This rating measures the schedule constraint imposed on the project team developing the software. Since the schedule considers mandatory deadlines, there should not be any compression or expansion for the whole project (*i.d. 100% of nominal schedule)*, therefore this parameter is set to **NOMINAL**.

Cost Drivers summary In Table 21 is reported the value of each cost driver as determined in this section and the corresponding effort multiplier as specified in Table 15.

Cost Driver	Value	Numerical Value (*Effort Multiplier*)
PERS	Nominal	1.00
RCPX	High	1.33
RUSE	Nominal	1.00
PDIF	Nominal	1.00
PREX	Low	1.22
FCIL	High	0.87
SCED	Nominal	1.00

Table 21: Cost drivers effort multipliers

2.2.3 Effort equation

The COCOMO II effort equation can now be computed as

$$\begin{aligned} PM &= A \cdot Size^{E} \cdot \prod_{i \in CostDrivers} C_i \\ &= 2.94 \cdot Size^{1.0761} \cdot (1 \cdot 1.33 \cdot 1 \cdot 1 \cdot 1.22 \cdot 0.87 \cdot 1) \\ &= 2.94 \cdot Size^{1.0761} \cdot 1.4112 \end{aligned}$$

Therefore

$$PM_{avg} = 2.94 \cdot 7.728^{1.0761} \cdot 1.4112 = 37$$

$$PM_{max} = 2.94 \cdot 11.256^{1.0761} \cdot 1.4112 = 56$$

2.2.4 Schedule estimation

According to the model definition, the duration D (in months) of the project can be estimated as

$$D = 3.67 \cdot (PM_{NS})^{F} \cdot \frac{SCED\%}{100}$$

where

- PM_{NS} is the Person-Months estimation without considering the SCED cost driver
- $F = 0.28 + 0.2 \cdot (E - 0.91)$
- E is the aggregation of scale factors previously determined
- $SCED\%$ is the SCED nominal value percentage, in our case is set to 100% (see SCED paragraph)

Since the effort multiplier of SCED cost driver is 1, it results

$$PM_{NS} = PM$$

Since as previously specified $E = 1.0761$ we can compute

$$F = 0.28 + 0.2 \cdot (1.0761 - 0.91) = 0.31322$$

Therefore

$$\begin{aligned} D_{avg} &= 3.67 \cdot 37^{0.31322} \cdot 1 = 11.373 \text{ months} \\ D_{max} &= 3.67 \cdot 56^{0.31322} \cdot 1 = 12.949 \text{ months} \end{aligned}$$

3 Schedule

In this section the schedule for the *PowerEnJoy* project will be presented; it is a high-level general schedule that allows the reader to understand the main phases of the project specification and development.

3.1 Notes for the charts

3.1.1 RASD Gantt chart (Figure 1)

- **Meetings with client**: all the resources will be present at the meeting
- **Modelization of the World and the Machine**: this part requires discussion and cooperation, hence it will be assigned to all the resources
- **Identification of use cases**: Davide will start working on this activity in date 28/10/2016
- **In progress meeting with client**: all the resources will be present at the meeting
- **Requirements refinement**: the requirements will be updated based on updates made in other parts of the document, during its development
- **Alloy modelization**: Davide will start working on this activity in date 05/11/2016
- **Document revision**: these days are left for the revision of the document and to complete tasks that eventually need more time

3.1.2 DD Gantt chart (Figure 2)

- **Architecture draft**: this part requires discussion and cooperation, hence it will be assigned to all the resources
- **Meeting with clients**: all the resources will be present at the meeting
- **Sequence diagrams**: Moreno will start working on this activity in date 29/11/2016

3.1.3 ITPD Gantt chart (Figure 3)

- **Integration test strategy and Definition of precedences**: these initial tasks will be accomplished at the same time and since they require discussion and cooperation, they will be assigned to all the resources
- **Document revision**: Davide will start working on this activity in date 19/12/2016

3.1.4 Development Gantt chart (Figure 4)

All of the tasks in this chart refers to high level activities; more specific charts will be produces during the development phase.

ID	Task Name	Start	Finish	Duration
1	**RASD**	**17/10/2016**	**11/11/2016**	**20d**
2	**Meetings with client**	**17/10/2016**	**19/10/2016**	**3d**
3	Identification of goals and stakeholders	17/10/2016	17/10/2016	1d
4	Requirements elicitation	18/10/2016	19/10/2016	2d
5	**Modelization of the World and the Machine**	**20/10/2016**	**25/10/2016**	**4d**
6	Identification of domain assumptions	20/10/2016	25/10/2016	4d
7	Identification of system goals	20/10/2016	25/10/2016	4d
8	Identification of requirements	20/10/2016	25/10/2016	4d
9	Writing scenarios	26/10/2016	26/10/2016	1d
10	Alloy model draft	26/10/2016	28/10/2016	3d
11	Identification of needed data	27/10/2016	27/10/2016	1d
12	Identification of use cases	27/10/2016	28/10/2016	2d
13	In progress meeting with client	02/11/2016	02/11/2016	1d
14	**Document refinement**	**03/11/2016**	**07/11/2016**	**3d**
15	Requirements refinement	03/11/2016	07/11/2016	3d
16	Data model refinement	03/11/2016	04/11/2016	2d
17	Sequence diagrams for use cases	03/11/2016	07/11/2016	3d
18	Alloy modelization	03/11/2016	07/11/2016	3d
19	Document revision	08/11/2016	11/11/2016	4d

Figure 1: RASD Gantt chart

ID	Task Name	Start	Finish	Duration
1	**DD**	**14/11/2016**	**02/12/2016**	**15d**
2	**Architecture draft**	**14/11/2016**	**16/11/2016**	**3d**
3	High level system view definition	14/11/2016	16/11/2016	3d
4	Definition of system boundaries	14/11/2016	16/11/2016	3d
5	Main architecture decisions	14/11/2016	16/11/2016	3d
6	Meeting with clients	17/11/2016	17/11/2016	1d
7	Refining architecture choices	18/11/2016	21/11/2016	2d
8	Main components component diagram	22/11/2016	23/11/2016	2d
9	ER DB Model	24/11/2016	24/11/2016	1d
10	Components internal structure	24/11/2016	25/11/2016	2d
11	Components external interfaces	24/11/2016	25/11/2016	2d
12	Mapping components to goals	28/11/2016	28/11/2016	1d
13	Sequence diagrams	25/11/2016	30/11/2016	4d
14	Mockups	29/11/2016	30/11/2016	2d
15	Deployment diagram	01/12/2016	01/12/2016	1d
16	Algorithms desing	01/12/2016	01/12/2016	1d
17	Revision and requirement traceability	02/12/2016	02/12/2016	1d

Figure 2: DD Gantt chart

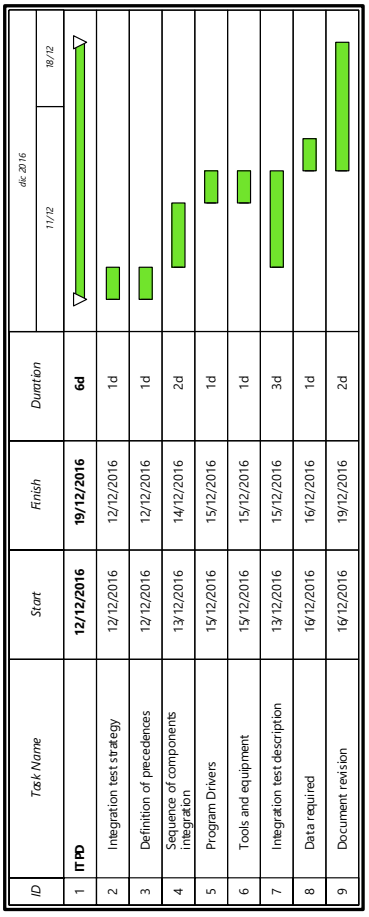


Figure 3: ITPD Gantt chart

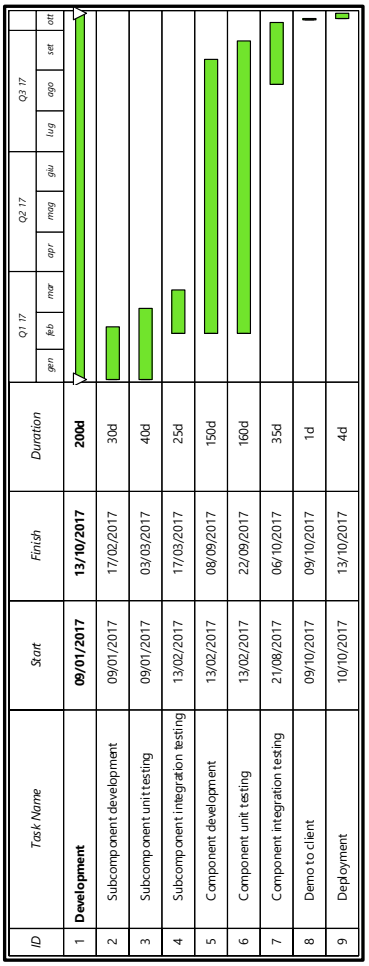


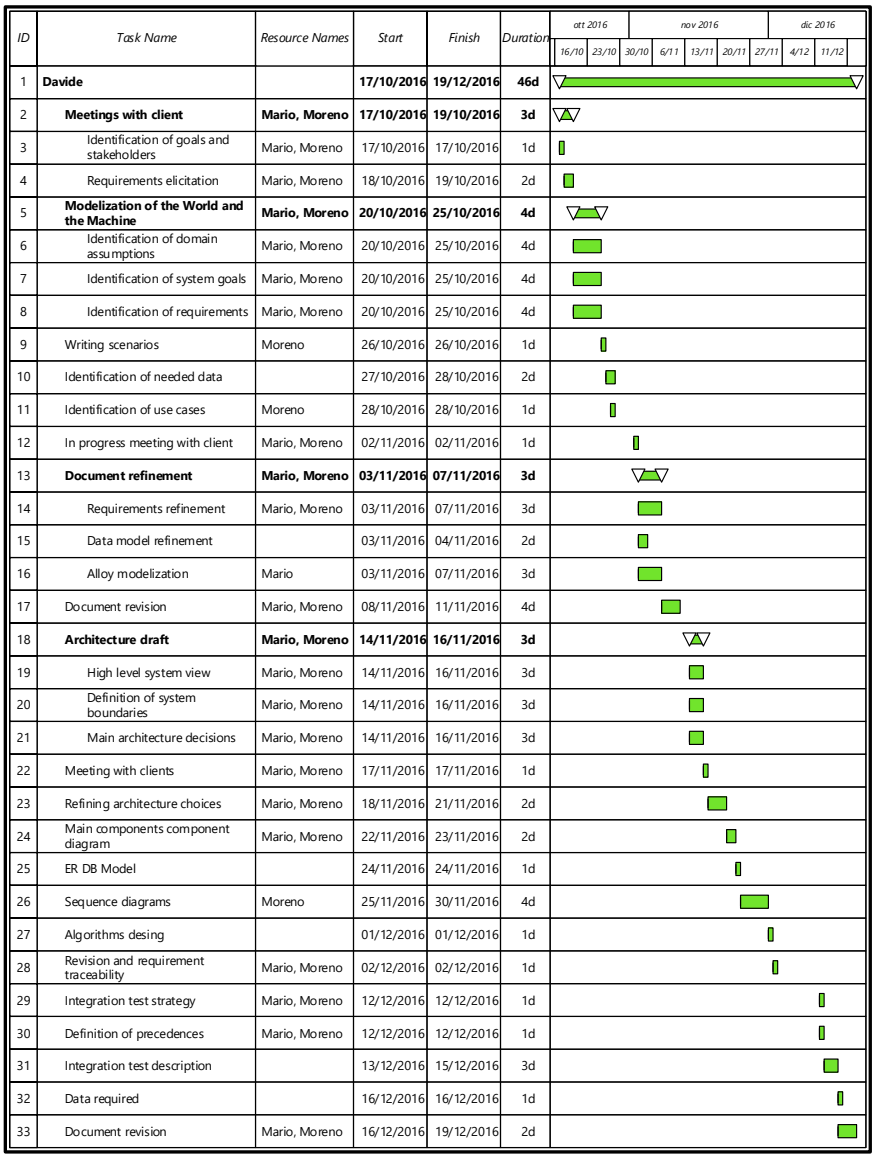
Figure 4: Development Gantt chart

4 Resources Allocation

In this section the resource allocation for the tasks related to *PowerEnJoy* project will be presented; it uses the high level tasks shown in the schedule section to describe which tasks will be assign to each of the team members.

4.1 Notes for the charts

- The duration of the first task of each diagram represents the number of working days in the interval of dates defined by the starting date of the first actual task assigned to the resource, and the ending date of the last one; in between this interval there may be days in which the resource has no assigned tasks. This was done to leave some elasticity to the project in order to avoid further delays in the project development.
- The *Resource Names* column contains the names of the team members who will collaborate with the resource in order to accomplish a given task. If this field is empty, it means that the resource will work on that task on his own.



ID	Task Name	Resource Names	Start	Finish	Duration
1	**Davide**		**17/10/2016**	**19/12/2016**	**46d**
2	**Meetings with client**	**Mario, Moreno**	**17/10/2016**	**19/10/2016**	**3d**
3	Identification of goals and stakeholders	Mario, Moreno	17/10/2016	17/10/2016	1d
4	Requirements elicitation	Mario, Moreno	18/10/2016	19/10/2016	2d
5	**Modelization of the World and the Machine**	**Mario, Moreno**	**20/10/2016**	**25/10/2016**	**4d**
6	Identification of domain assumptions	Mario, Moreno	20/10/2016	25/10/2016	4d
7	Identification of system goals	Mario, Moreno	20/10/2016	25/10/2016	4d
8	Identification of requirements	Mario, Moreno	20/10/2016	25/10/2016	4d
9	Writing scenarios	Moreno	26/10/2016	26/10/2016	1d
10	Identification of needed data		27/10/2016	28/10/2016	2d
11	Identification of use cases	Moreno	28/10/2016	28/10/2016	1d
12	In progress meeting with client	Mario, Moreno	02/11/2016	02/11/2016	1d
13	**Document refinement**	**Mario, Moreno**	**03/11/2016**	**07/11/2016**	**3d**
14	Requirements refinement	Mario, Moreno	03/11/2016	07/11/2016	3d
15	Data model refinement		03/11/2016	04/11/2016	2d
16	Alloy modelization	Mario	03/11/2016	07/11/2016	3d
17	Document revision	Mario, Moreno	08/11/2016	11/11/2016	4d
18	**Architecture draft**	**Mario, Moreno**	**14/11/2016**	**16/11/2016**	**3d**
19	High level system view	Mario, Moreno	14/11/2016	16/11/2016	3d
20	Definition of system boundaries	Mario, Moreno	14/11/2016	16/11/2016	3d
21	Main architecture decisions	Mario, Moreno	14/11/2016	16/11/2016	3d
22	Meeting with clients	Mario, Moreno	17/11/2016	17/11/2016	1d
23	Refining architecture choices	Mario, Moreno	18/11/2016	21/11/2016	2d
24	Main components component diagram	Mario, Moreno	22/11/2016	23/11/2016	2d
25	ER DB Model		24/11/2016	24/11/2016	1d
26	Sequence diagrams	Moreno	25/11/2016	30/11/2016	4d
27	Algorithms desing		01/12/2016	01/12/2016	1d
28	Revision and requirement traceability	Mario, Moreno	02/12/2016	02/12/2016	1d
29	Integration test strategy	Mario, Moreno	12/12/2016	12/12/2016	1d
30	Definition of precedences	Mario, Moreno	12/12/2016	12/12/2016	1d
31	Integration test description		13/12/2016	15/12/2016	3d
32	Data required		16/12/2016	16/12/2016	1d
33	Document revision	Mario, Moreno	16/12/2016	19/12/2016	2d

Figure 5: Davide Piantella tasks Gantt chart

ID	Task Name	Resource Names	Start	Finish	Duration
1	**Mario**		**17/10/2016**	**19/12/2016**	**46d**
2	**Meetings with client**	**Davide, Moreno**	**17/10/2016**	**19/10/2016**	**3d**
3	Identification of goals and stakeholders	Davide, Moreno	17/10/2016	17/10/2016	1d
4	Requirements elicitation	Davide, Moreno	18/10/2016	19/10/2016	2d
5	**Modelization of the World and the Machine**	**Davide, Moreno**	**20/10/2016**	**25/10/2016**	**4d**
6	Identification of domain assumptions	Davide, Moreno	20/10/2016	25/10/2016	4d
7	Identification of system goals	Davide, Moreno	20/10/2016	25/10/2016	4d
8	Identification of requirements	Davide, Moreno	20/10/2016	25/10/2016	4d
9	Alloy model draft		26/10/2016	28/10/2016	3d
10	In progress meeting with client	Davide, Moreno	02/11/2016	02/11/2016	1d
11	**Document refinement**	**Davide, Moreno**	**03/11/2016**	**07/11/2016**	**3d**
12	Requirements refinement	Davide, Moreno	03/11/2016	07/11/2016	3d
13	Alloy modelization		03/11/2016	07/11/2016	3d
14	Document revision	Davide, Moreno	08/11/2016	11/11/2016	4d
15	**Architecture draft**	**Davide, Moreno**	**14/11/2016**	**16/11/2016**	**3d**
16	High level system view	Davide, Moreno	14/11/2016	16/11/2016	3d
17	Definition of system boundaries	Davide, Moreno	14/11/2016	16/11/2016	3d
18	Main architecture decisions	Davide, Moreno	14/11/2016	16/11/2016	3d
19	Meeting with clients	Davide, Moreno	17/11/2016	17/11/2016	1d
20	Refining architecture choices	Davide, Moreno	18/11/2016	21/11/2016	2d
21	Main components component diagram	Davide, Moreno	22/11/2016	23/11/2016	2d
22	Components external interfaces		24/11/2016	25/11/2016	2d
23	Mapping components to goals	Moreno	28/11/2016	28/11/2016	1d
24	Deployment diagram	Moreno	01/12/2016	01/12/2016	1d
25	Revision and requirement traceability	Davide, Moreno	02/12/2016	02/12/2016	1d
26	Integration test strategy	Davide, Moreno	12/12/2016	12/12/2016	1d
27	Definition of precedences	Davide, Moreno	12/12/2016	12/12/2016	1d
28	Sequence of components integration	Moreno	13/12/2016	14/12/2016	2d
29	Program Drivers		15/12/2016	15/12/2016	1d
30	Document revision	Davide, Moreno	16/12/2016	19/12/2016	2d

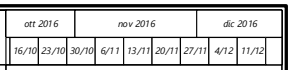


Figure 6: Mario Scrocca tasks Gantt chart

ID	Task Name	Resource Names	Start	Finish	Duration
1	**Moreno**	**Davide, Mario**	**17/10/2016**	**19/12/2016**	**46d**
2	**Meetings with client**	**Davide, Mario**	**17/10/2016**	**19/10/2016**	**3d**
3	Identification of goals and stakeholders	Davide, Mario	17/10/2016	17/10/2016	1d
4	Requirements elicitation	Davide, Mario	18/10/2016	19/10/2016	2d
5	**Modelization of the World and the Machine**	**Davide, Mario**	**20/10/2016**	**25/10/2016**	**4d**
6	Identification of domain assumptions	Davide, Mario	20/10/2016	25/10/2016	4d
7	Identification of system goals	Davide, Mario	20/10/2016	25/10/2016	4d
8	Identification of requirements	Davide, Mario	20/10/2016	25/10/2016	4d
9	Writing scenarios	Davide	26/10/2016	26/10/2016	1d
10	Identification of use cases		27/10/2016	28/10/2016	2d
11	In progress meeting with client	Davide, Mario	02/11/2016	02/11/2016	1d
12	**Document refinement**	**Davide, Mario**	**03/11/2016**	**07/11/2016**	**3d**
13	Requirements refinement	Davide, Mario	03/11/2016	07/11/2016	3d
14	Sequence diagrams for use cases		03/11/2016	07/11/2016	3d
15	Document revision	Davide, Mario	08/11/2016	11/11/2016	4d
16	**Architecture draft**	**Davide, Mario**	**14/11/2016**	**16/11/2016**	**3d**
17	High level system view	Davide, Mario	14/11/2016	16/11/2016	3d
18	Definition of system boundaries	Davide, Mario	14/11/2016	16/11/2016	3d
19	Main architecture decisions	Davide, Mario	14/11/2016	16/11/2016	3d
20	Meeting with clients	Davide, Mario	17/11/2016	17/11/2016	1d
21	Refining architecture choices	Davide, Mario	18/11/2016	21/11/2016	2d
22	Main components component diagram	Davide, Mario	22/11/2016	23/11/2016	2d
23	Components internal structure		24/11/2016	25/11/2016	2d
24	Mapping components to goals	Mario	28/11/2016	28/11/2016	1d
25	Sequence diagrams	Davide	29/11/2016	30/11/2016	2d
26	Deployment diagram	Mario	01/12/2016	01/12/2016	1d
27	Revision and requirement traceability	Davide, Mario	02/12/2016	02/12/2016	1d
28	Integration test strategy	Davide, Mario	12/12/2016	12/12/2016	1d
29	Definition of precedences	Davide, Mario	12/12/2016	12/12/2016	1d
30	Sequence of components integration	Mario	13/12/2016	14/12/2016	2d
31	Tools and equipment		15/12/2016	15/12/2016	1d
32	Document revision	Davide, Mario	16/12/2016	19/12/2016	2d

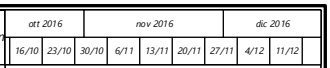


Figure 7: Moreno Raimondo Vendra tasks Gantt chart

5 Risk management

This section describe possible risks for the *PowerEnJoy system* and some possible reactive and proactive strategies to mitigate them.

Note Probability level in ascending order is:

1. *Low*
2. *Moderate*
3. *High*

Dangerousness level of effects in ascending order is:

1. *Trivial*
2. *Moderate*
3. *Serious*
4. *Catastrophic*

5.1 Project risks

Failure to achieve a deadline

Description	Various contingencies could bring to fail to achieve deadlines.
Probability	Moderate
Effects	Moderate
Actions	Considering in the project schedule all the issues that may arise during the project development is definitely a difficult challenge; allocating some extra time after each major phase of the project would help to avoid the shift of some tasks on the schedule because of a missed deadline.

Personnel shortfall

Description	Some member of the development team may leave the project; since the size of the team is quite small it may cause serious effects.
Probability	Low
Effects	Serious
Actions	In order to prevent serious effects the project must be well documented in each phase, meetings must be organized to allow each member of the group to have a general knowledge of each part of the project and critic tasks must not be assigned to a single team member.

Change of requirements	
Description	Requirements may need some changes for different reasons: not exhaustive requirements, misunderstood requirements or missing requirements.
Probability	Moderate
Effects	Moderate/Serious
Actions	To avoid the need to change requirements it would be really useful to start interacting with all the project stakeholders since the beginning of the project development (for example also allowing some users to interact with some sorts of mockups of the application during the requirements design phase). Changing requirements during the development phase may create the necessity to change already implemented software and/or software design. The relevance on the project of these changes must be evaluated. If the issue is serious and depends on misunderstandings with the customer during the requirements discussion, it could be necessary to re-discuss project contract, schedule and estimated budget. Anyway a modular and reusable thinking of the software architecture would help to be more reactive to these kinds of situations.

5.2 Technical risks

Underestimation of requests' load	
Description	Traffic and requests are much more than what has been estimated, resulting in high latency and possibly in service unavailability
Probability	Low
Effects	Serious
Actions	Analyze the traffic through all component, consider an upgrade of hardware components, external API contracts and traffic plans. During the development use a modular software design to manage multiple requests in parallel and to improve scalability.

Data Loss	
Description	Data of the system may be compromised due to an hardware fault or to wrong writing operations.
Probability	Low
Effects	Catastrophic
Actions	It is convenient to store an updated backup copy of data possibly on a different location from the system database one.

Wrong external component specification	
Description	An external component is not compliant to its specification
Probability	Low
Effects	Serious
Actions	Try to find a solution with the component's owner, if impossible find a different compatible component and consider taking legal actions.

Difficulties during development phase	
Description	Some members of the development team may experience difficulties in the development process of some specific features of the system due to the lack of previous experience.
Probability	Low
Effects	Moderate
Actions	It may be required to ask consulting to external companies.

Car supplier changes	
Description	The company for some reason may need to change the car supplier. Since our system relies on the car embedded system provided by our supplier it would be a considerable risk if the the communication protocol used with cars changed.
Probability	Low
Effects	Serious
Actions	To mitigate the possible effects of this situation the system software must abstract from the type of communication protocol used with cars and interface components must be designed to be as reusable as possible in order to reduce to minimum needed changes.

External APIs bad behaviour	
Description	The system needs to interact with some external APIs that may not behave as expected.
Probability	Low
Effects	Moderate
Actions	External APIs interaction must be deeply tested during the development phase in order to identify possibly unexpected behaviours in this phase: the company that developed the API could be contacted to eventually ask to fix the behaviour or, if it is related to an open source project, a pull request may be proposed to fix the behaviour.

External APIs downtime	
Description	The external APIs the system has to interact with may be unreachable.
Probability	Low/Moderate
Effects	Serious
Actions	Always try to provide as many functionality as possible: if the failure is not in a core functionalities, e.g. payments API fails, it would be useful to develop the project such that it is possible to keep the whole system online and operative and at the same time store the information about unprocessed payments for future proceeding. When it is possible take into account the usage of redundant APIs according to failure recovery paradigms.

5.3 Business risks

Competitors	
Description	Competitors saturate the market with their car sharing services
Probability	High
Effects	Serious
Actions	Make some market surveys in order to determine and implement some *killer features* to persuade users to use the *PowerEnJoy* service.

Financial crisis	
Description	Customer may have financial troubles during the development of the system causing a reduction of the budget.
Probability	Low
Effects	Catastrophic
Actions	A good study of feasibility would detect this risk. This situation could be mitigated by meeting the customer and re-discussing features and agreed requirements; this discussion could also lead to the end of the project development.

Appendices

A Software and tools used

For the development of this document we used

- LaTeX as document preparation system
- Git & GitHub as version control system
- Microsoft Visio and Excel as Gantt diagrams generators

B Hours of work

This is the amount of time spent to redact this document:

- Davide Piantella: $\sim$ 18 hours
- Mario Scrocca: $\sim$ 16 hours
- Moreno R. Vendra: $\sim$ 14 hours

References

[1] D. Piantella, M. Scrocca, M.R. Vendra, *PowerEnJoy: Requirements Analysis and Specification Document*, Politecnico di Milano - Software Engineering II Project, 2016

[2] D. Piantella, M. Scrocca, M.R. Vendra, *PowerEnJoy: Design Document*, Politecnico di Milano - Software Engineering II Project, 2016

[3] D. Piantella, M. Scrocca, M.R. Vendra, *PowerEnJoy: Integration Test Plan Document*, Politecnico di Milano - Software Engineering II Project, 2017

[4] Quantitative Software Management, Function Point Languages Table version 5.0, QSM SLOC/FP Data, http://www.qsm.com/resources/function-point-languages-table

[5] International Function Point Users Group (IFPUG), *Function Point Counting Practices Manual*, Release 4.2

[6] University of Southern California, Center for Systems and Software Engineering, *COCOMO II*, http://csse.usc.edu/csse/research/COCOMOII/cocomo_main.html

[7] University of Southern California, Center for Systems and Software Engineering, *COCOMO II.2000 Model Definition Manual*, http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf