Politecnico di Milano

A.Y. 2016-2017

Software Engineering II: “PowerEnJoy”

Project Plan Document

version 1.0

January 22, 2017

Pozzati Edoardo (mat. 809392), Stefanetti Nadia (mat. 810622)

Table of Contents

1 Introduction

1.1 Purpose and Scope

This document represents the Project Plan Document for PowerEnJoy.
Its main purpose is to provide a reasonable estimate of the expected complexity of PowerEnJoy and assist the project leader in the delicate phase of cost and effort estimation.
This information will therefore serve as a guideline to define the required budget, the resources allocation and the schedule of the activities.
In the second section of the document, *Function Points Analysis (FPA)* is going to be used in order to estimate the size, while *COCOMO* is going to be applied to provide an estimate of the effort and cost in terms of number of lines of source code (SLOC) to be written and the average effort required for the development process itself.
In the third section, the organizational structure of our project plan and a possible schedule will be identified, in the interest of covering all activities within the deadline from the requirements identification to the implementation and testing activities, and with a proper redistribution of tasks among the team members.
In the fourth section, the resources (all members of our development group) will be allocated to the various tasks.
Finally, we are going to define the all the possible risks that the project could face throughout its life-cycle, from the development to the deployment phase, with their relevance and the associated recovery actions

2 Function Points Estimation

This section is specifically devoted to provide some estimations of the effort needed in designing and coding the project for PowerEnJoy. In order to obtain a reasonable estimate of the size, the Function Points approach will be used, taking into account all the main functionalities of PowerEnJoy and estimating the correspondent amount of lines of code to be written in Java.

2.1 Internal Logic Files (ILFs)

An Internal Logic File (ILF) is defined as a homogeneous set of data used and managed by the application being developed.
The ILFs of our system match closely the database tables and are the following:

- User
- Registration
- Ride
- Reservation
- Operator
- Car
- Safe Area

2.2 External Interface Files (EIFs)

An External Interface File (EIF) is defined as a homogeneous set of data used by the application but generated and maintained elsewhere.
The only EIF of the system consists in the interface with:

- GoogleMapsAPI
- OnBoardApplication

2.3 External Inputs (EIs)

An External Input (EI) is defined as an elementary operation to elaborate data coming from the external environment.
The EIs of the system are the following:

- User registration

- User login
- User profile management
- Operator login
- Car Status
- Car Sensors

2.4 External Outputs (EOs)

An External Output (EO) is defined as an elementary operation that generates data for the external environment. It usually includes the elaboration of data from logic files.
The EOs of the system are the following:

- Notification to User
- Notification to Operator
- Generated Password
- Confirmation credentials (Email, License, CreditCard)
- Set Car Status

2.5 External Inquiries (EQs)

An External Inquiry (EQ) is defined as an elementary operation that involves input and output, without significant elaboration of data from logic files.
The EQs for the system are the following:

- Operator request
- Search car
- Compute fare
- Insert position
- Creation e deletion of reservation

2.6 Complexity levels of Function Points

Table 1.1 shows how to evaluate the weights of Function Points, based on the number of data elements.

	Data Elements		
Record Elements	1-19	20-50	51+
1	Low	Low	Avg
2-5	Low	Avg	High
6+	Avg	High	High

a) Weight estimation for ILFs and EIFs.

	Data Elements		
Record Elements	1-5	6-19	20+
0-1	Low	Low	Avg
2-3	Low	Avg	High
4+	Avg	High	High

b) Weight estimation for EOs and EQs.

	Data Elements		
Record Elements	1-4	5-15	16+
1	Low	Low	Avg
2-3	Low	Avg	High
3+	Avg	High	High

c) Weight estimation for EIs.

Table 1.1: Estimation of weights for different types of Function Points.

2.7 Weights of Function Points

The scores associated to all types of function points, by type and weight, are shown in Table 1.3.

2.8 UFP to SLOC conversion

The multiplicator to convert the Unadjusted Function Points to Source Lines of Code for Java is **53**.

2.9 Count of FPs, by type and weight

The total count of Function Points of our system is shown in Table 1.5.

Functions	**Weights**
User	Low
Registration	High
Ride	Avg
Reservation	High
Operator	Avg
Car	High
SafeArea	Avg

a) Computed weights for ILFs.

Functions	**Weights**
GoogleMapsAPI	Avg
OnBoardApplication	High

b) Computed weights for EIFs.

Functions	Weights
User registration	Avg
User login	Low
User profile management	Low
Operator login	Low
Car Status	Low
Car Sensors	Avg

c) Computed weights for EIs.

Functions	Weights
Notification to User	Low
Notification to Operator	Low
Generated Password	Low
Confirmation credentials	Low
Set Car Status	Low

d) Computed weights for EOs.

Functions	Weight
Operator request	Low
Search car	Avg
Compute fare	Avg
Insert position	Low
Creation e deletion reservation	Avg

e) Computed weights for EQs.

Table 1.2: The evaluation of the weights of the functions according to Table 1.1.

Function Type	Complexity-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 1.3: Scores of function points by type and weight.

Function	**Type**	**Complexity**	**Weight**
User	ILF	Low	7
Registration	ILF	High	15
Ride	ILF	Avg	10
Reservation	ILF	High	15
Operator	ILF	Avg	10
Car	ILF	High	15
SafeArea	ILF	Avg	10
GoogleMapsAPI	EIF	Avg	7
OnBoardApplication	EIF	High	10
User registration	EI	Avg	4
User login	EI	Low	3
User profile management	EI	Low	3
Operator login	EI	Low	3
Car Status	EI	Low	3
Car Sensors	EI	Avg	4
Notification to User	EO	Low	4
Notification to Operator	EO	Low	4
Generated Password	EO	Low	4
Confirmation credentials	EO	Low	4
Set Car Status	EO	Low	4
Operator request	EQ	Low	3
Search car	EQ	Avg	4
Compute fare	EQ	Avg	4
Insert position	EQ	Low	3
Creation e deletion reservation	EQ	Avg	4

Table 1.4: Weights computed for all the functions.

Function Type	**Complexity-Weight**			**Total Points**
	Low	Average	High	
External Input	4 x 3	2 x 4	0 x 6	20
External Output	5 x 4	0 x 5	0 x 7	20
External Inquiry	2 x 3	3 x 4	0 x 6	18
Internal Logic File	1 x 7	3x 10	3 x 15	82
External Interface File	0 x 5	1 x 7	1 x 10	17
Total Points	45	57	55	**157**
SLOC	X53			**8321**

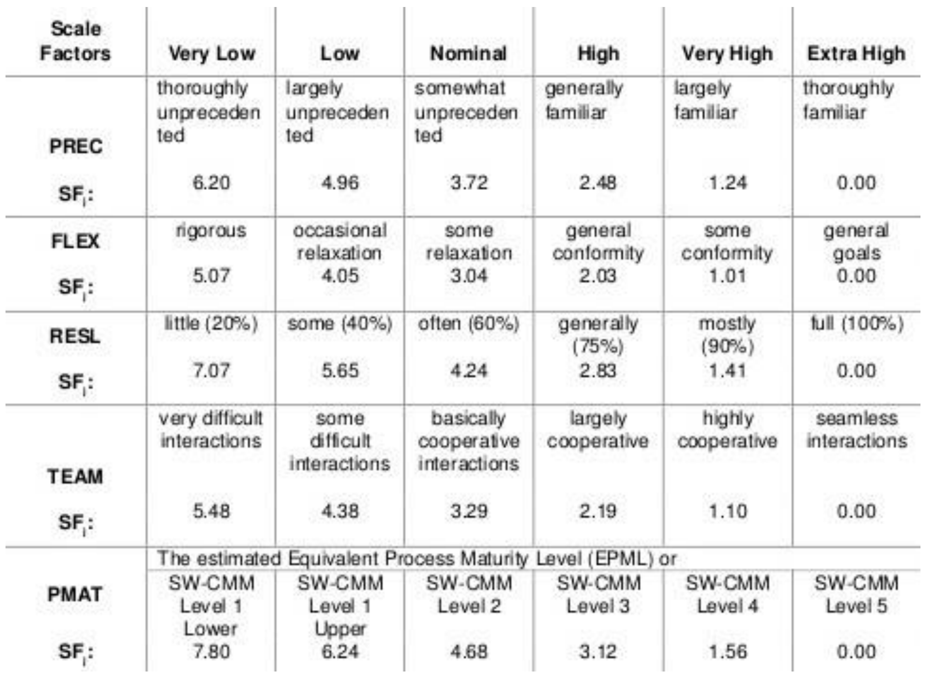
Table 1.5: Count of Function Points of our system.

3 COCOMO II estimation

In this section the COCOMO II approach is going to be used to estimate the cost and effort required to develop PowerEnJoy.

3.1 Scale Drivers

Scale Drivers are the parameters that reflect the non-linearity of the effort with relation to the number of SLOC. They describe the project itself and determinate the exponent of the Effort Equation (Equation 2.1). Each scale driver has a range of rating levels from Very Low to Extra High, each with its own rate.



Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	thoroughly familiar
SF_j:	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
SF_j:	5.07	4.05	3.04	2.03	1.01	0.00
RESL	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
SF_j:	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
SF_j:	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	The estimated Equivalent Process Maturity Level (EPML) or					
	SW-CMM Level 1 Lower	SW-CMM Level 1 Upper	SW-CMM Level 2	SW-CMM Level 3	SW-CMM Level 4	SW-CMM Level 5
SF_j:	7.80	6.24	4.68	3.12	1.56	0.00

Figure 3.1: Scale Factor Values for COCOMO II Models.

PREC Precedentedness
It represents the previous experience of the team with the organization and development of large scale projects. Very Low means no previous experience, Extra High means that the organization is completely familiar with this application domain. We assume that the precedentedness is *Low* because most of the notions used in this project are new to us.

FLEX Development flexibility
It is an indicator of the degree of flexibility in the development process with respect to the given specification and requirements. Very Low means a prescribed process is used; Extra High means that the client only sets general goals. We set it to *Nominal* because the project is bounded to a set of prescribed specifications, but a certain level of flexibility in the definition of the requirements and in the design process is allowed.

RESL Risk resolution
It points out the extent of risk analysis carried out, therefore the level of awareness and reactiveness with respect to said risks. Very Low means little analysis, Extra High means a complete and thorough risk analysis. We set it to *High*, since we will offer a rather detailed risk analysis in Section 6.

TEAM Team cohesion
This value is correlated to how well the development team members know each other and work together in a cooperative way. Very Low means very difficult interactions, Extra High means an integrated and effective team with no communication problems. We set it to *Very High*, since the cohesion among the two of us is optimal and the team is effective with no communication problems.

PMAT Process maturity
This parameter reflects the process maturity of the organization, describing how well the behaviours, practices and processes of an organization can reliably produce required outcomes. It is set at *High*, which corresponds to CMM Level 3: “It is characteristic of processes at this level that there are sets of defined and documented standard processes established and subject to some degree of improvement over time. These standard processes are in place (i.e., they are the AS-IS processes) and used to establish consistency of process performance across the organization”.
The estimated scale drivers for our project, together with the formula to compute the exponent E, are shown in Table 3.1.

Code	**Name**	**Factor**	**Value**
PREC	Precedentedness	Low	4.96
FLEX	Development flexibility	Nominal	3.04
RESL	Risk resolution	High	2.83
TEAM	Team cohesion	Very High	1.10
PMAT	Process maturity	High	3.12
Total	$E = 0.91 + 0.01 \times \sum_i SF_i$		1.0605

Table 3.1: Scale Drivers for our project.

3.2 Cost Drivers

Cost Drivers are the parameters of the Effort Equation that reflect some characteristics of the developing process and act as multiplicators on the effort needed to build the project. They appear as factors in the Effort Equation (Equation 3.1).
The estimated cost drivers for our project, together with the formula to compute the EAF (Effort Adjustment Factor), are shown in Table 3.19.

RELY Required Software Reliability
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 reliability is Low. If a failure would risk human life then reliability is Very High.
Our value will be *Nominal* because a downtime would not lead to any critical consequences besides some inconvenience to users but would bring moderate financial losses.

RELY Descriptors	Slightly in-convenience	Easily re-coverable losses	Moderate re-coverable losses	High financial loss	Risk to human life	
Rating Levels	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Effort Multipliers	0.82	0.92	1.00	1.10	1.26	n/a

Table 3.2: RELY Descriptors.

DATA Data Base Size
This values try to estimate the effects that large databases could have in our product development. The reason the size of the database is important to consider it because of the effort required to generate the test data that will be used to test the program.
The value will be *Very High* due to our strong dependence with tests database.

DATA Descriptors		D/P < 10	10 ≤ D/P ≤ 100	100 ≤ D/P ≤ 1000	D/P > 1000	
Rating Levels	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Effort Multipliers	n/a	0.90	1.00	1.14	1.28	n/a

Table 3.3: DATA Descriptors.

CPLEX Product Complexity
In accordance with the new COCOMO II CPLEX rating scale, this value will be *Nominal* since no complex algorithms are involved and I/O processing includes status checking and error processing.

Rating Levels	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Effort Multipliers	0.73	0.87	1.00	1.17	1.34	1.74

Table 3.4: CPLEX Cost Driver.

RUSE Required Reusability
This cost driver accounts for the additional effort needed to construct components intended for reuse on the 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.
We've chosen a *Nominal* value because the architecture is quite dependent to

the goals required and the re-usability is limited to the project itself.

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 3.5: RUSE Descriptors.

DOCU Documentation match to life-cycle needs
Several software cost models have a cost driver for the level of required documentation. In COCOMO II, the rating scale for the DOCU cost driver is evaluated in terms of the suitability of the project's documentation to its life-cycle needs. The rating scale goes from Very Low (many life-cycle needs uncovered) to Very High (very excessive for life-cycle needs).
In our case, every need of the product life-cycle is already foreseen in the documentation, so this parameter is set to *Nominal* since the team does not want to spend excessive time on documentation, but a right-sized one is very important to reduce system maintenance costs and improve the software understanding.

DOCU Descriptors	Many life-cycle needs uncovered	Some life-cycle needs uncovered	Rightsized to life-cycle needs	Excessive for life-cycle needs	Very excessive for life-cycle needs	
Rating Levels	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Effort Multipliers	0.81	0.91	1.00	1.11	1.23	n/a

Table 3.6: DOCU Descriptors.

ACAP Analyst Capability
Analysts are personnel that work on requirements, high level design and detailed design. The major attributes that should be considered in this rating are analysis and design ability, efficiency and thoroughness, and the ability to communicate and cooperate. It is set to *Nominal* since we do not have any previous experience in finding requirements, but we think the analysis of the problem has been conducted in a thorough and complete way with respect to a potential real world implementation.

ACAP Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating Levels	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Effort Multipliers	1.42	1.19	1.00	0.85	0.71	n/a

Table 3.7: ACAP Descriptors.

PCAP Programmer Capability
This value it's set to *Nominal*, because our programming abilities are in the average.

PCAP Descriptors	15th percentile	35th per-centile	55th per-centile	75th per-centile	90th per-centile	
Rating Levels	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Effort Multipliers	1.34	1.15	1.00	0.88	0.76	n/a

Table 3.8: PCAP Descriptors.

PCON Personnel Continuity
The rating scale for Continuity is in terms of the project’s annual personnel turnover: from 3%, very high, to 48%, very low.
We’ve selected a value *Very High* since the turnover is totally absent.

PCON Descriptors	48% / year	24% / year	12% / year	6% / year	3% / year	
Rating Levels	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Effort Multipliers	1.29	1.12	1.00	0.90	0.81	n/a

Table 3.9: PCON Descriptors.

APEX Application Experience
This rating is dependent on the level of applications experience of the project team developing the software system or subsystem.
Our value is *Low* because we have some experience in the development of Java applications, but we never tackled a JEE system of this kind before.

APEX Descriptors	$\leq$ 2 months	6 months	1 year	3 years	6 years	
Rating Levels	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Effort Multipliers	1.22	1.10	1.00	0.88	0.81	n/a

Table 3.10: APEX Descriptors.

PLEX Platform Experience
The Post-Architecture model broadens the productivity influence of platform experiences, recognizing the importance of understanding the use of more powerful platforms, including more graphic user interface, database, networking, and distributed middleware capabilities.
As said before, because of our little experience about platforms as databases, UI, client/server architecture, this value is set to *Low*.

PLEX Descriptors	$\leq$ 2 months	6 months	1 year	3 years	6 years	
Rating Levels	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Effort Multipliers	1.19	1.09	1.00	0.91	0.85	n/a

Table 3.11: PLEX Descriptors.

LTEX Language and Tool Experience
This is a measure of the level of programming language and software tool experience of the project team developing the software system or subsystem. For the same reasons described before, also this value is set to *Low*.

LTEX Descriptors	≤ 2 months	6 months	1 year	3 years	6 years	
Rating Levels	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Effort Multipliers	1.20	1.09	1.00	0.91	0.84	n/a

Table 3.12: LTEX Descriptors.

TIME Execution Time Constraint
This is a measure of the execution time constraints 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 PowerEnJoy will offer a service that is expected to be extensively and frequently used by a large customer base, TIME is set to *High*.

TIME Descriptors			≤ 50% use of available execution time	70% use of available execution time	85% use of available execution time	95% use of available execution time
Rating Levels	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Effort Multipliers	n/a	n/a	1.00	1.11	1.29	1.63

Table 3.13: TIME Descriptors.

STOR Main Storage Constraint
This rating represents the degree of main storage constraints imposed on a software system or subsystem. The service provided by the system is supposed to be repeatedly and regularly used. On the other hand, modern technologies can offer a lot of storage capacity in order to prevent saturation, therefore STORE is set to *High*.

STOR Descriptors			≤ 50% use of available execution time	70% use of available execution time	85% use of available execution time	95% use of available execution time
Rating Levels	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Effort Multipliers	n/a	n/a	1.00	1.05	1.17	1.46

Table 3.14: STOR Descriptors.

PVOL Platform Volatility
It represents the volatility of the platform. The system will not require frequent major updates, but minor changes, patches and bug fixes are planned to be release constantly in time. Our estimation is that this is a stable system with low

volatility. *Low* is a good choice here.

PVOL Descriptors		Major: 12 months Minor: 1 month	Major: 6 months Minor: 2 weeks	Major: 2 months Minor: 1 weeks	Major: 2 weeks Minor: 2 days	
Rating Levels	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Effort Multipliers	n/a	0.87	1.00	1.15	1.30	n/a

Table 3.15: PVOL Descriptors.

TOOL Use of Software Tools
Software tools that will be used in the project are strong, mature, moderately integrated and able to manage the life cycle of the software-to-be, therefore the rating level is set to *High*.

TOOL Descriptors	Edit, code, debug	Simple, frontend, backend CASE, little integration	Basic lifecycle tools, moderately integrated	Strong, mature life-cycle tools, moderately integrated	Strong, mature, proactive lifecycle tools, well integrated with processes, methods, reuse	
Rating Levels	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Effort Multipliers	1.17	1.09	1.00	0.90	0.78	n/a

Table 3.16: TOOL Descriptors.

SITE Multisite Development
Although we live in two different cities, this value is set to *Very High*, due to our strong cooperation through the usage of chats, emails, phone calls and other internet services.

SITE Collo-cation Descriptors	Intern-ational	Multi-city and multi-company	Multi-city or multi-company	Same city or metro area	Same building or complex	Fully collocated
SITE Com-munications Descriptors	Some phone, mail	Individual phone, FAX	Narrow band email	Wideband electronic communi-cation	Wideband elect comm, occasional video conf	Interactive mulimedia
Rating Levels	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Effort Multipliers	1.22	1.09	1.00	0.93	0.86	0.80

Table 3.17: SITE Descriptors.

SCED Required Development Schedule
It measures the schedule constraints imposed on the project team developing

the system-to-be. It is set to *High* since although our efforts were well distributed over the available development time, the definition of all the required documentation took a consistent amount of time, especially for the requirement analysis and the design phases.

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 3.18: SCED Descriptors.

Code	**Name**	**Factor**	**Value**
RELY	Required Software Reliability	Nominal	1.00
DATA	Data Base Size	Very High	1.28
CPLEX	Product Complexity	Nominal	1.00
RUSE	Required Reusability	Nominal	1.00
DOCU	Documentation match to life-cycle needs	Nominal	1.00
ACAP	Analyst Capability	Nominal	1.00
PCAP	Programmer Capability	Nominal	1.00
PCON	Personnel Continuity	Very High	0.81
APEX	Applications Experience	Low	1.10
PLEX	Platform Experience	Low	1.09
LTEX	Language and Tool Experience	Low	1.09
TIME	Execution Time Constraint	High	1.11
STOR	Main Storage Constraint	High	1.05
PVOL	Platform Volatility	Low	0.87
TOOL	Use of Software Tools	High	0.90
SITE	Multisite Development	Very High	0.86
SCED	Required Development Schedule	High	1.00
Total	$EAF = \prod_i C_i$		1.0634

Table 3.19: Cost Drivers for our project.

3.3 Effort

Now, having both cost drivers product and scale drivers factors we can compute the effort, in Person-Month (PM) with the following equation (3.1):

$$Effort = A \times EAF \times KSLOC^{E} \qquad (3.1)$$

Where:

- $EAF = \prod_i C_i$: derived from the cost drivers analysis.

(= 1.0634)

- $E = 0.91 + 0.01 \times \sum_i SF_i$: derived from the scale drivers analysis.

  (= 1.0605)

- *KSLOC*: Kilo-Source Lines of Code, derived from Fps analysis.

  (= 8.321)

- $A = 2.94$ (constant for COCOMO II)

The total effort results in **29.6 person-months**.

3.4 Duration

$$Duration = 3.67 \times (Effort)^{0.28 + 0.2 \times (E - 0.91)} \qquad (3.2)$$

Where:

- *Effort* is the effort as calculated in Equation 3.1.

  (= 29.6)

- *E* is the exponent depending on the scale drivers analysis (the same one of the effort equation). (= 1.0605)

The duration calculated from Equation 3.2 results in a total of **10.5 months**. This would result in 2.8 developers needed for this project.

Since our group consists of only 2 people, a reasonable development time would be:

$$\frac{29.6 \text{ pm}}{2 \text{ people}} = 14.8 \text{ months}$$

In order to be cautious with the task scheduling, we will approximate the project overall time to **15 months**.

4 Task and Schedule

The main tasks involving this project are:

1. Deliver the **Requirement Analysis and Specification Document (RASD)**, containing the goals, the domain assumptions and the functional and non-functional requirements of the software system.

2. Deliver the **Design Document (DD)**, containing the description of the architecture and of the design of the software system-to-be.

3. Deliver the **Integration Testing Plan Document (ITPD)**, containing the strategy used to perform integration testing on the components of the system.

4. Deliver the **Project Plan Document (PPD)**, which is this 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. Prepare a brief **presentation** (15 min) about the delivered documents, with slides.

6. **Implementation** of the software system and writing of unit tests.

7. Perform **integration testing** on the whole system.

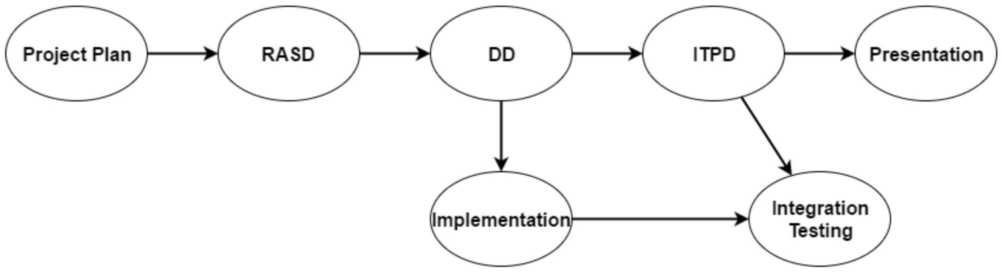


Figure 4.1 Directed Acyclic Graph showing the dependencies among tasks.

From the COCOMO estimation performed in Section 3, we expect the entire project to last 15 months, so it will be presumably finished by Dicember 2017.

N.	Activity	Start Date	Deadline
1	RASD	17/10/2016	13/11/2016
2	DD	21/11/2016	11/12/2016
3	ITPD	02/01/2017	15/01/2017
4	PP	16/01/2017	22/01/2017
5	Presentation	06/02/2017	12/02/2017
6	Implementation	13/02/2017	05/12/2017
7	Integration Testing	06/12/2017	19/12/2017

Tab 4.1: Schedule for project tasks.

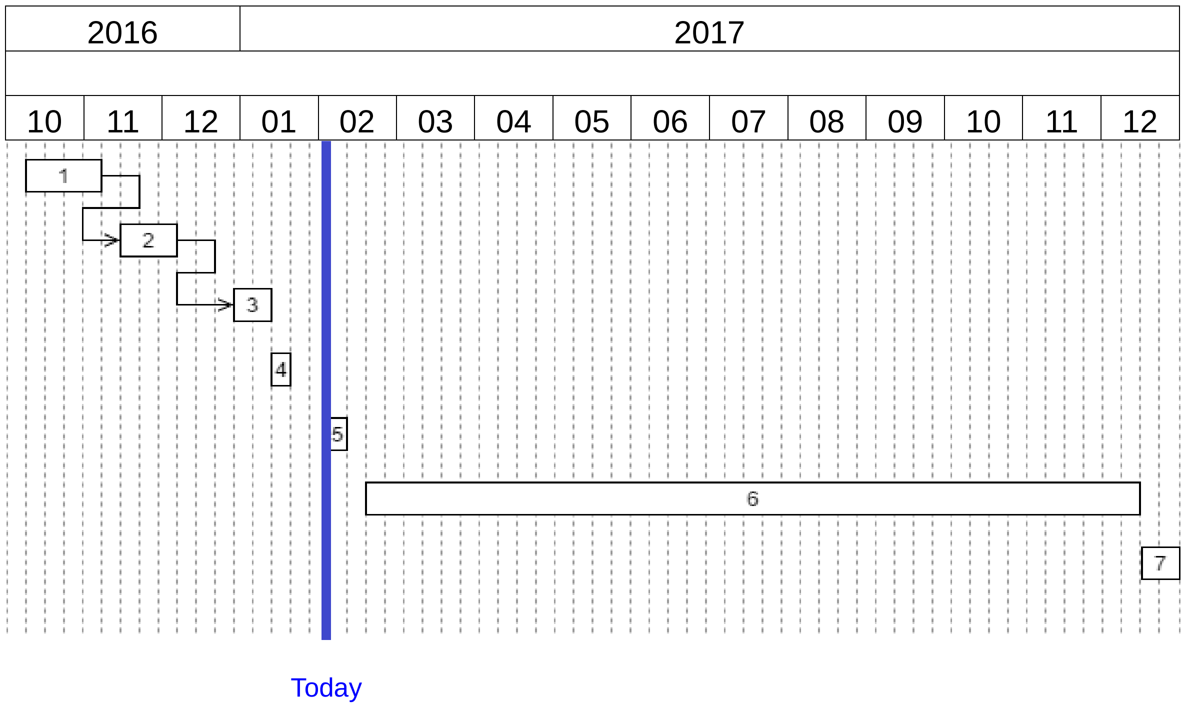


Figure 4.2: Gantt chart of the project.

5 Resource Allocation

This section is meant to show the way our available resources are allocated to the project.
We will provide a general overview of how the tasks defined by the schedule in the previous section will be divided between the two members of the development team.
It does not aim to be a very specific and refined schedule of the single development phases, in fact the tasks are grouped with a high level of abstraction and resources are assigned to these high-level tasks. This is because micromanaging the work of people on a large project is of little use.
The resulting schedule gives to all the team members a general overview on the whole project: every assigned task involves the contribution of each member.
This enlarges slightly the time needed to complete the project, but also increase the overall awareness of every member of the team. It also enables a more accurate control on the work of each other in order to reduce the possibility to create misunderstandings.
The documents to be written related to the project are divided into arguments which mainly reflect the division in sections; each argument will be first discussed together.
Regarding the implementation and the integration testing, each member of the team is asked to focus on a tier of the application; the unit tests related will be written and executed by the other team member, in order to make the testing process more efficient and accurate.
The division of the work is elaborated according to the schedule (Table 4.1, Figure 4.2).
The division of work between team members is shown in tables 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7.

Resources	**17/10/16 to 13/11/16**			
	1st week	2nd week	3rd week	4th week
Nadia Edoardo	Spec req Intro	Overall descr Spec req	System features System features	Revision Revision

Table 5.1: Resources allocation for RASD.

Resources	**21/11/16 to 11/12/16**		
	1st week	2nd week	3rd week
Nadia Edoardo	Architectural design Architectural design	Arch design / UI design Arch design / Algorithm	Architectural revision Architectural revision

Table 5.2: Resources allocation for DD.

Resources	02/01/17 to 15/01/17	
	1st week	2nd week
Nadia Edoardo	Integration strategy Individual steps	Individual steps Revision

Table 5.3: Resources allocation for ITPD.

Resources	16/01/2017 to 22/01/2017
	1st week
Nadia Edoardo	COCOMO / Risk / Revision Funcion Points / Task Schedule / Resource Allocation

Table 5.4: Resources allocation for Planning.

Resources	06/02/2017 to 12/02/2017
	1st week
Nadia Edoardo	Slide Slide

Table 5.5: Resources allocation for Presentation.

Resources	13/02/2017 to 05/12/17				
	1st month	2nd month	3rd month	4th month	
Nadia Edoardo	Web Tier Database	Web Tier Database	Business Tier Business Tier	Test Database Mobile Client	
	5th month	6th month	7th month	8th month	9th month
Nadia Edoardo	Mobile Client Mobile Client	Test Database Test Web Tier	Test Business Test Web Tier	Test Database Test Business	Revison Revision

Table 5.6: Resources allocation for Implementation.

Resources	06/12/2017 to 19/12/2017	
	1st week	2nd week
Nadia Edoardo	Business – Data Business – Web	Client – Web Mobile – Business

Table 5.7: Resources allocation for Integration Testing.

6 Risk Management

In this section we are going to assess the main risks that the project development may face. Some of them could pose technical issues, while others are related with political or financial challenges.
The PowerEnJoy system project is associated with many risks, which have to be considered in a complete project planning, owing to their uncertain ad dangerous nature. A sudden change in mind, actions, and economical situations and alike could drift the project development into failure; this is the reason why they are here analysed, following a specific approach that splits them up into three groups: project, technical and business risks.
In the following sections we will provide a detailed description of the risks and the countermeasures. Moreover, the 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, Medium, High, Very High*. The effects of a risk upon occurrence can be classified as: *Negligible, Marginal, Critical, Catastrophic.*

6.1 Project risks

These are the risks that threaten the project plan. If they become real, it is likely that the project schedule will slip and that costs will increase.
The project risks are listed below. An evaluation of those risks is provided in Table 6.1.

Delays over the expected deadlines: due to several overlapping tasks the team is involved into, or due to the fact that key members of the team may get ill just prior to important milestones or meetings, or may be ill for prolonged periods of time, the project is very likely to require more time than expected.
A strict organization among the team components is fundamental. This implies a constant cooperation between developers, in order to squeeze even the tiniest time slots available for this project.
Nonetheless, if schedule delays happens, we may release a first, incomplete but working version (e.g. without the web interface or the plugins for profile management and reservation deletion) and build the less essential features later.
Also, we have to consider the possibility of people quitting the company, as the IT job market is quite flexible. A possible solution for this problem is to split duties and responsibilities across multiple people, so that no single person is in charge of a specific task.

Lack of estimations/schedules: a lack of experience in this area can lead to serious errors in evaluating development time.
We suggest to deal with this problem by studying previous works on a similar subject.

Lack of communication among team members: the team often works remotely and this can lead to misunderstandings in fundamental decisions, to conflicts over the division of the work among team members and to conflicts in code versioning. Collaboration issues can sometimes be crucial, especially when dealing with task divisions.
Those difficulties can be overcome by explicitly defining (e.g. in this document) the responsibilities of each team member, and by writing clear and complete specification and design documents and by arranging frequent brief meetings in order for each member to gain awareness of the level of progress achieved.

Lack of experience in programming with the specific frameworks: the team has no actual experience in programming using Java EE. This will certainly slow down the development.
Another risk might come from overestimating the knowledge of a specific matter or programming technique that our programmers and engineers have.
Adding people to the project should not be seen as the primary solution here, unless the task is extremely specific. A good antidote is to hire knowledgable and flexible people beforehand.

Requirements change: the requirements may change during the development in unexpectedly. A sudden growth in requirements can lead to a rush to meeting deadlines, jeopardizing the overall quality.
This kind of risk cannot be prevented, but can be mitigated by writing reusable and extensible software. Also, thinking with a broader mind on the first stages can be very helpful; however, the team should be careful against over-engineering (which can also paralyze the development).

Risk	**Probability**	**Effects**
Delays	Medium / High	Marginal
Lack of estimations/schedules	Very Low	Critical
Lack of communication	Very Low	Negligible
Lack of experience	Very High	Marginal
Requirements change	Medium	Marginal

Table 6.1: Evaluation of project risks.

6.2 Technical risks

These risks involve the actual implementation of the project and can threaten the quality and timeliness of the software to be developed. Because our system is quite complex and it is composed by different components, the integration among them may be a problem. That’s why the integration test is a essential part in our project.
The technical risks are listed below. An evaluation of those risks is provided in Table 6.2.

Integration testing failure: after the implementation of some components, they may not pass the integration testing phase: this can require to rewrite large pieces of software.
This risk can be mitigated by defining precisely the interfaces between

components and subsystems: all components must be unit tested as soon as possible, to eliminate serious bugs when they first appear. Afterwards, integration tests must be done as soon as possible using stubs and drivers according to the specifications contained in [3].

Downtime: the system could go down for excessive load, software bugs, hardware failures or power outages. This risk can be mitigated by building redundant systems and placing them in geographically separate data centres and by performing testing at all levels. In cases in which this is not possible, structuring the code to make it more reliable, robust and maintainable is always a good countermeasure.

Scalability issues: if the servers happen to be unreliable or in the case the system could not scale properly with the increasing number of users, a work of major redesign will be carried out.
A possible plan to reduce the risk is to use a cloud infrastructure from a third-party provider to host our system and to provide a scalable design of the overall architecture, both in software and in hardware choices.

Unreadable code: with the growth of the project, the code may become overloaded, badly structured and unreadable.
Documenting the code can be a reasonable way to mitigate this risk. Furthermore, writing a good Design Document before starting to code and performing code inspection periodically can come in handy too.

Data loss: data can be lost because of hardware failures, misconfigured software or deliberate attacks. A loss of the whole source code, or significant parts thereof, would be a disaster.
This problem can be prevented by enforcing a reliable backup plan. Backups should be distributed over multiple, redundant locations and should be kept in a separate place from the system.

Data leaks: the application may be susceptible to security issues if not well-designed. Misconfiguration, software bugs and deliberate attacks can expose the users' personal data.
All modern standards in computer security guidelines must be followed, especially when dealing with the user input, which has to be correctly verified and processed, and industry security standards must be adopted, by encrypting communications and by doing regular penetration testing.

Dependency on external services and components: a change in the terms and conditions of the Mapping Service, or even just a modification of the API itself, could pose serious financial or technical problems. We are somewhat more protected as for database and message broker technology, as there is a greater number of vendors and the access methods are more or less standardized.
A possible countermeasure is to design the code to be as portable as possible and with a great modularity and independence between components, exploiting the information hiding principle to the fullest.

Risk	**Probability**	**Effects**
Integration testing failure	Medium	Critical
Downtime	Low	Marginal
Scalability issues	Low	Critical
Unreadable code	Medium	Marginal
Data loss	Low	Catastrophic
Data leaks	Low	Catastrophic
Dependency on external services and components	Low	Critical

Table 6.2: Evaluation of technical risks.

6.3 Business risks

These risks involve the company developing the software. This may cause troubles to the whole software product threatening its viability. (e.g. if the business cannot subsidize the software being developed anymore).
The economical risks are listed below. An evaluation of those risks is provided in Table 6.3.

Additional costs: testing devices & infrastructure (PCs, several mobile phones, server rent) need to be purchased and configured. This is going to increase costs that may be not sustainable if the company is too small.
We suggest to clearly define testing tools as we did in [2] and [3], in order to avoid worthless spending.

Bankruptcy: the income from the usage of the application may not be enough to support development and maintenance of the software system, causing the company to find itself in serious financial trouble.
In order to avoid bad investments, we suggest to use surveys and a good feasibility study. That strategy allows us to better understand trend of the market and so to know whether our application could be interesting for people. In that way it could be possible estimate an average profit also.
Another market strategy is about the feedback usage. We suggest to provide to the software a kind of mechanism which allows users and operators to evaluate the software goodness.

Competitors: other companies can build equivalent or better products at a lower price and put PowerEnJoy out of the market. The competitiveness of our product must be continuously enhanced by introducing innovative features.

Risk	**Probability**	**Effects**
Additional costs	Medium	Critical
Bankruptcy	Medium	Catastrophic
Competitors	High	Marginal

Table 6.3: Evaluation of economical risks.

Appendix A: Used Tools

A.1 Microsoft Word 2013

To redact and format this document.

A.2 *g*it

To submit this document in the online repository.

A.3 Dropbox

Used as version control system in order to lead development.

A.4 Draw.io

To create and design the provided diagrams.

Appendix B: Hours of work

This is the time spent by each group member in order to redact this document:

- Pozzati Edoardo: ~10 hours
- Stefanetti Nadia: ~10 hours
- Total work time: ~20 hours

Acronyms

API	Application Programming Interface.
COCOMO	COnstructive COst MOdel.
DB	Data Base.
DBMS	Data Base Management System.
DD	Design Document.
EI	External Input.
EIF	External Interface File.
ELF	External Logic File.
EO	External Output.
EQ	External Inquiry.
FP	Function Point.
FPA	Function Point Analysis.
GPS	Global Positioning System.
ILF	Internal Logic File.
ITPD	Integration Test Plan Document.
JEE	Java Enterprise Edition.
RASD	Requirements Analysis and Specification Document.
SLOC	Source Lines Of Code.

Bibliography

The indications provided in this document are based on the ones stated in the previous deliverables for the project:

[1] A.Y. 2016/2017 Software Engineering 2 - *Requirements Analysis and Specification Document* - Pozzati Edoardo, Stefanetti Nadia.

[2] A.Y. 2016/2017 Software Engineering 2 - *Design Document* - Pozzati Edoardo, Stefanetti Nadia.

[3] A.Y. 2016/2017 Software Engineering 2 – *Integration Test Plan Document* - Pozzati Edoardo, Stefanetti Nadia.

Moreover it is based on the specifications concerning the RASD assignment [4]

[4] Luca Mottola and Elisabetta Di Nitto, *Software Engineering 2: Project goal, schedule and rules*, 2016.

[5] COCOMO II - Model Definition Manual, Version 2.1, 1995 – 2000, Center for Software Engineering, USC: http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf

[6] *Function Points Analysis Training Course*: www.SoftwareMetrics.com Longstreet Consulting Inc