Project Plan

Li Xiaoxu	10545608
Jia Hongyan	10470662
Lang Shuangqing	10517902

Contents

Chapter 1

Introduction

1.1 Revision History

Version of this document: 1.2
Last update: 20/01/2017

Version 1.1 typo fixed
Version 1.2 re-calculated time in chapter 4

1.2 Purpose and Scope

This PP document defining a software and management planning for the PowerEnjoy project, building a visualization of the practical project development, in order to take timely and effective measure when the actual implementation of the project deviates the software plan.

The purpose of this document is to ensure the project carrying out following the the plan, to set the monitoring standards of the project's implementation, to reduce the project risk and to successfully complete the project.

In the first section, we introduce the purpose and scope, and reference documents, and some basic background of the PP.

In the second section, we're going to use the Function Points and CO-COMO approaches together to provide an estimate of the expected size of PowerEnjoy in terms of lines of code and of the cost/effort required to actually develop it.

In the third section, we'll reuse these figures to propose a possible schedule for the project that covers all activities from the requirements identification to the implementation and testing activities.

In the forth section we're going to assign the different members of our development group to the various tasks.

Finally, we're going to elaborate on the possible risks that PowerEnjoy could face during the various phase of the project and provide some general conclusions.

1.3 List of Definitions and Abbreviations

Name	**Definition**
RASD	Requirements Analysis and Specification Document
DD	Design Document
ITPD	Integration Test Plan Document
PP	Project Plan
DB	Database
EP	Function Points
ILF	Internal Logic File
ELF	External Logic File
EI	External Input.
EO	External Output
EQ	External Inquiries
ETA	Estimated Time of Arrival
DBMS	Data Base Management System
API	Application Programming Interface
APP	Mobile Application, here implement on IOS, Android system

Table 1.1 List of Definitions and Abbreviations

1.4 List of Reference Documents

This document refers to the following documents:

[1] AA 2016---2017 Software Engineering 2- Project goal, schedule, and rules
[2] *Li Xiaoxu, Lang Shuangqing, Jia Hongyan,* Software Engineering 2: PowerEnjoy- Requirements Analysis and Specification Document

[3] *Jia Hongyan, Lang Shuangqing,* Software Engineering 2: PowerEnjoy- Design Document
[4] *Chitti Eleonora, De Nicolao Pietro, Delbono Alex,* Software Engineering 2: myTaxiService- Integration Test Plan Document
[5] *Van Vliet, H., Van Vliet, H., & Van Vliet, J. C. (1993)*. Software engineering: principles and practice (Vol. 3).
[6] *Casati Fabrizio, Castelli Valerio,* Software Engineering 2: myTaxiService - Project Plan Document
[7] 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*

Chapter 2

Project size, cost and effort estimation

This section is specifically focused on providing the estimations of the expected size, cost and required effort of PowerEnjoy.

Function Points approach has been applied for the size estimation, taking into account all the main functionalities of PowerEnjoy and estimating the correspondent amount of lines of code to be written in Java. This estimation will only take into account the parts of the project that concur to the implementation of the business logic and will disregard the aspects concerning the user interface.

COCOMO approach has been applied for the cost and effort estimation, as an initial guidance of the amount of lines of code computed with the FP approach.

2.1 Function Points estimation

The Function Points approach provides an estimation of the size of a project taking the amount of functionalities to be developed and their complexity as input.

The estimation is based on the usage of figures obtained through statistical analysis of real projects, which have been properly normalized and condensed in the following Tables 2.1 and Table 2.2, and Table 2.3 is the overall FPs computed for all the function.

	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 2.1 Estimation of weights for Function Points

Function Type	**Complexity-Weight**		
	Simple	Medium	Complex
External Input	3	4	6
External Output	4	5	7
External Inquiry	3	4	6
Internal Logic File	7	10	15
External Interface File	5	7	10

Table 2.2 Scores of function points by type and weight

2.1.1 Internal Logic Files (ILFs)

An Internal Logic File (ILF) is defined as a homogeneous set of data used and managed by the application.

The ILFs of our system match closely the database tables and are the following:

- Car
- SafeArea
- Data
- Queue
- Reservation
- Fee
- Price
- Cost
- Profit

Car: the system has to store information about cars, which will be a single table that holds code, number of car' plate, capacity, together status of in use, status of reservation and the location (latitude and longitude) of the car.

SafeArea: pre-defined simple table with information of safe areas for parking cars, containing the position (longitude and latitude) and the region.
Data: is used for system backup, so it is a list of data highly condensed data of the system.

Reservations: stored in a dedicated table that holds all the information about the reservation ID, the cellphone number of the passenger who booked them, and the related code of the car.

Queue: a data structure of a queue with head and tail as pointer. It is used to stored time information for computing algorithm 3.1 described in DD document.

Fee: stored in a simple table for store basic fee.

Price: stored in a dedicated table stored the price, computing by algorithm 3.2 described in DD document.

Cost: stored in a dedicated table representing the overall cost of the system for financial purpose.

Profit: stored in a dedicated table representing the overall profit of the system for financial purpose.

ILF	**Complexity**	**FPs**
Car	Low	7
SafeArea	Low	7
Data	High	15
Queue	Average	10
Reservation	Low	7
Fee	Low	7
Price	Low	7
Cost	Low	7
Profit	Low	7
Total		74

2.1.2 External Interface File

An External Interface File (EIF) is defined as a homogeneous set of data used by the application but generated and maintained by other applications.

The first EIF of the system consists in the interface with Google Maps, described in the DD [3]. We identify this EIF as **Map** in following tables.

The second EIF consists in the interface with SMS Gateway, we identify this EIF as **SMS** in following tables.

EIF	**Complexity**	**FPs**
Map	High	10
SMS	Low	5
Total		15

2.1.3 External Input

An External Input (EI) is defined as an elementary operation to elaborate data coming from the external environment.

- User registration
- User profile management
- User login
- User logout
- Car reservation
- Insert passenger number
- Select car-sharing option
- Delete reservation
- View reservation history
- Pay the bill

EI	Complexity	FPs
User registration	Average	4
User profile management	Average	4
User login	Low	3
User logout	Low	3
Car reservation	High	6
Insert passenger number	Low	3
Select car-sharing option	Low	3
Delete reservation	Low	3
View reservation history	Low	3
Pay the bill	Average	4
Total		36

2.1.4 External Output (EO)

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.

- Providing password
- Notification of reservation

- Notification of assigned car's position
- Providing Time counting-down
- Notification of reservation expired
- Notification of assigned car
- Notification of low battery
- Notification of bill
- Confirmation of payment
- Providing Time counting-down

EO	Complexity	FPs
Providing password	Low	4
Notification of reservation	Low	4
Notification of assigned car's position	Average	5
Providing Time counting-down	Low	4
Notification of reservation expired	Low	4
Notification of assigned car	Low	4
Notification of low battery	Low	4
Notification of bill	Low	4
Confirmation of payment	Low	4
Providing Time counting-down	Low	4
Total		41

2.1.5 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.

- Ride sharing
- An administrator can retrieve the full list of cars, users.
- An administrate can retrieve the financial condition (profit and cost).

EQ	Complexity	FPs
Ride sharing	High	6
Retrieve list of cars	Low	3
Retrieve list of users	Low	3
Retrieve profit	Low	3
Retrieve cost	Low	3
Total		18

2.1.6 Overall estimation

Function	Type	Complexity	FPs
Car	ILF	Low	7
SafeArea	ILF	Low	7
Data	ILF	High	15
Queue	ILF	Average	10
Reservation	ILF	Low	7
Fee	ILF	Low	7
Price	ILF	Low	7
Cost	ILF	Low	7
Profit	ILF	Low	7
Map	ELF	High	10
SMS	ELF	Low	5
User registration	EI	Average	4
User profile management	EI	Average	4
User login	EI	Low	3
User logout	EI	Low	3
Car reservation	EI	High	6
Insert passenger number	EI	Low	3
Select car-sharing option	EI	Low	3
Delete reservation	EI	Low	3
View reservation history	EI	Low	3
Pay the bill	EI	Average	4
Providing password	EO	Low	4
Notification of reservation	EO	Low	4
Notification of assigned car's position	EO	Average	5
Providing Time counting-down	EO	Low	4
Notification of reservation expired	EO	Low	4
Notification of assigned car	EO	Low	4
Notification of low battery	EO	Low	4
Notification of bill	EO	Low	4
Confirmation of payment	EO	Low	4
Providing Time counting-down	EO	Low	4
Ride sharing	EQ	High	6
Retrieve list of cars	EQ	Low	3
Retrieve list of users	EQ	Low	3
Retrieve profit	EQ	Low	3
Retrieve cost	EQ	Low	3

Table 2.3 FPs computed for all the function

Function Type	Value
Internal Logic Files	74
External Interface Files	15
External Inputs	36
External Outputs	41
External Inquires	18
Total	184

Table 2.4 Value of different function types

Considering Java Enterprise Edition as a development platform, we can estimate the total number of lines of code. According to section 2.3 in [7], the UFP to SLOC Conversion Ratios we select is 53.

$$SLOC = 184 * 53 = 9752$$

2.2 COCOMO II estimation

2.2.1 Scale Drivers, Cost Drivers,Effort

In this section we're going to use the COCOMO II approach to estimate the cost and effort needed and for evaluating the values of the scale drives, we refer to the following official COCOMO II table in Table 2.5 :

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprecede-nted	largely unpreced-ented	somewhat unpreceden-ted	generally familiar	largely familiar	thoroughly familiar
SFj	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	Rigorous	Occasion-al relaxation	Some Relaxation	General Conformity	Some conformity	General Goals
SFj	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%)
SFj	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	very difficult	some diffi- cult	basically cooperative	largely cooperati	highly cooperati	seamless interaction

SFj	interactions 5.48	interactions 4.38	interactions 3.29	ve 2.19	ve 1.10	s 0.00
PMAT	Level 1 Lower	Level 1 Upper	Level 2	Level 3	Level 4	Level 5
SFj	7.80	6.24	4.68	3.12	1.56	0.00

Table 2.5 official COCOMO II table

A brief description for each scale driver:

• Precedentedness: it reflects the previous experience of our team with the development of large scale projects. Since we have less experience in this field, this value will be low.

• Development flexibility: it reflects the degree of flexibility in the development process with respect to the external specification and requirements. So there are very strict requirements on the functionalities but nothing specific is stated as for the technology to be used, this value will be low.

• Risk resolution: reflects the level of awareness and reactiveness with respect to risks. The risk analysis we performed can be controlled so the value will be set to Nominal.

• Team cohesion: it's an indicator of how well the team members know each other and work together in a cooperative way. For our team, the value is very high.

• Process maturity: although we had met some problems during the development of the project, the goals have been successfully achieved. Since this is our first project of this kind, this value is set to level 3.

The results of our evaluation is the following:

Scale Driver	Factor	Value
Precedentedness (PREC)	Low	4.96
Development flexibility (FLEX)	Nominal	3.04

Risk resolution (RESL)	Nominal	4.24
Team cohesion (TEAM)	Very high	1.10
Process maturity (PMAT)	High	3.12
Total		16.46

$$\text{Effort} = A \times EAF \times KSLOC^{E} \qquad (2.1)$$

Where:

- $EAF = \prod_i C_i$ product of all cost about drivers
- $E = 0.91 + 0.01 \times \prod_i SF_i$: depends on the scale drivers.
- KSLOC: Kilo-Source Lines of Code, as estimated in chapter 2.1.
- $A = 2.94$

The total effort results in 32 person-months

Code	Name	Factor	Value
PREC	Precedentedness	Low	4.96
FLEX	Development flexibility	Nominal	3.04
RESL	Risk resolution	Nominal	4.24
TEAM	Team cohesion	Very High	1.10
PMAT	Process maturity	High	3.12
TOTAL	$E=0.91+0.01 \times \sum_i SF_i$		1.0746

Table 2.6 Scale Drivers for our project.

Code	**Name**	**Factor**	**Value**
RELY	Required Software Reliability	Nominal	1.00
DATA	Data base size	Nominal	1.00
CPLX	Product Complexity	Nominal	1.00
RUSE	Required Reusability	High	1.07
DOCU	Documentation match to life-cycle needs	Nominal	1.00
TIME	Execution Time Constraint	Nominal	1.00
STOR	Main Storage Constraint	Nominal	1.00
PVOL	Platform Volatility	Low	0.87
ACAP	Analyst Capability	High	0.85
PCAP	Programmer Capability	Nominal	1.00
APEX	Application Experience	Very low	1.22
PLEX	Platform Experience	Very low	1.19
LTEX	Language and Tool Experience	Low	1.09
PCON	Personnel Continuity	Very high	0.81
TOOL	Usage of Software Tools	Nominal	1.00
SITE	Multisite Development	High	0.93
SCED	Required Development Schedule	High	1.00
Total	$EAF = \prod_i C_i$		0.943

Table 2.7: Cost Drivers for our project.

2.2.2 Duration

$$\text{Duration} = 3.67 \times (\text{PM})\ 0.28+0.2\times(E-0.91) \quad (2.2)$$

The duration is 10.93 months

Where:

• PM is the effort as calculated in Equation 2.1.

• E is the exponent depending on the scale drivers (the same one of the effort equation).

Since our group consists of 3 people, a reasonable development time would be:

$$32\text{pm} / 3\ \text{people} = 10.6\ \text{months}$$

In order to be cautious with the task scheduling, we will approximate it to 11 months.

Chapter 3

Tasks and schedule

The main tasks involving this project are:

1. Deliver the Requirement Analysis and Specification Document, containing the goals, the domain assumptions, and the functional and nonfunctional requirements of the software system.

2. Deliver the Design Document, containing the architecture and the design of the software system.

3. Deliver the Integration Testing Plan Document, containing the strategy used to perform integration testing on the system.

4. Deliver the Project Plan, which is this document.

5. Prepare a brief presentation (~ 15 min) about the delivered documents, with slides.

6. Implement the software system and write unit tests.

7. Perform integration testing on the system.

We need to note that, as new requirements can emerge, new choices are made and the development goes on, the process can be iterated multiple times. In particular, unit and integration testing will be continuously performed throughout the development process.

However, some tasks need to be concluded before some other can begin: the dependency graph for the tasks is shown in Figure 3.1.

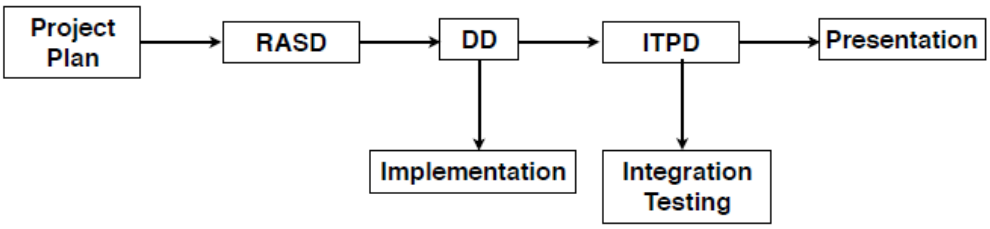

Figure 3.1: Directed Acyclic Graph showing the dependencies among tasks

There are no fixed deadlines, instead, for the development of the software. From the COCOMO estimation performed in chapter 2, we expect the entire project to last 11 months, so it will be presumably finished by August 2017. The schedule for our project is outlined in Table 3.1.

Activity	Start Date	Deadline
RASD	2016-10.16	2016-11.13
DD	2016-11.15	2016-12.11
ITPD	2016-12.13	2017-1.15
Project Plan	2017-1.16	2017-1.22
Presentation	2017-1.23	2017-2.5
Implementation	2017-2.8	2017-7.2
Integration Testing	2017-7.3	2017-8.20

Table 3.1: Schedule for project tasks.

Chapter 4

Resources allocation

This chapter has the purpose to show how available resources are allocated to the project. Because of the project's property, the resources cannot be material. The whole project is completely abstract. It is divided into different parts such as RASD and DD etc. So the resources are related with the tasks. It means that getting the tasks, getting the resources.

As for tasks, they are allocated to three team members. Different members take care of different parts. But before start, team members would tackle the problems together. The next few tables would show the actual allocation of different parts of project.

Allocation for RASD

	Jia Hongyan	Lang Shuangqing	Li Xiaoxu
1 Intro		×	×
2 Actor identifying			×
3 Requirements	×	×	
4 Scenario identifying		×	
5 Uml models			×
6 Alloy modeling	×		
7 Future development		×	
8 Used tools	×	×	×
9 Hours of work		×	
10 Changelog		×	

Allocation for DD

	Jia Hongyan	Lang Shuangqing	Li Xiaoxu
1 Intro		×	

2 Architecture Design	×		Not participate because of illness
3 Algorithm Design		×	
4 User Interface Design		×	
5 Requirements traceability	×		
6 Effort spent	×		
7 References		×	
8 Used tools	×	×	
9 Changelog		×	

Allocation for ITPD

	Jia Hongyan	Lang Shuangqing	Li Xiaoxu
1 Intro		×	Not participate because of illness
2 Integration Strategy	×		
3 Individual Steps and Test Description		×	
4 Tools and Test Equipment Required	×	×	
5 Program Stubs and Test Data Required	×		
6 Effort spent		×	
7 Used tools	×	×	
8 Bibliography		×	
9 Changelog	×		

Allocation for PPD

	Jia Hongyan	Lang Shuangqing	Li Xiaoxu
1 Intro	×		
2 Project size, cost and effort estimation	×	×	
3 Schedule		×	
4 Resources allocation			×
5 Risks			×
Appendix			×

These tables above show the allocation of finished parts. Otherwise, next few steps will be solved in similar ways.

As for project, we plan to complete the whole project within one year

(09.2016~08.2017) starting from paper work to programming. First few months, the completely background research would take place, and finish the relevant documents. Then programming should be on schedule. Finally, there will be integration testing. The following table would show more details.

Allocation of project

Events	Timestamp
Project starting	1st month
Group organization	1st month
First meeting	1st month
Background research（RASD）	1st month
Second meeting	2nd month
Initially design（DD）	2nd month
Third meeting	3rd month
Testing design（ITPD）	3rd month
Fourth meeting	4th month
Complete the plan（PP）	4th month
Start programming	5th month
Debugging	6th month
Function analyzing	6th month
Continue programming	6th month
Debugging	8th month
Finish programming	9th month
Integration testing	10th month
Complete project	11th month

Chapter 5
Risks Managment

In this section we're 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. At different time it would face different problems.

The first risk should be this project itself. It can also be defined by individual problem. This may happen during the design phase. There are some realistic problems. As time goes by, the team members may face some problems which they cannot solve easily because of poor experience in programming. Besides, the requirements would change by the development of the project, because human cannot consider everything before doing something. These problems get a very high probability in happening. These will cause delay of the project or even worse. But these kind of problems can be mitigated by a good preparation. This way could effectively lower the risks' probability.

The second risk would be the technical problem. After the implementation of some components, they may not pass the integration testing phase. And the system could go down for excessive load, software bugs, hardware failures or power outages. Also with the growth of the project, the code maybe come overloaded, badly structured and unreadable. These problems above also have a high probability. These risks can be mitigated by doing integration tests early using stubs and drivers, or by performing code inspection periodically. And writing a good Design Document before starting to code is very necessary. There is another situation in these kind of risk. It is data loss caused by unexpected hardware failures. This situation can make the system broken. To prevent this, during the coding phase, making several backups is a real high-level operation.

The third risk is about financial and political. As we all know, if this kind

of threat really happens, it is deathful to the project. Because this problem cannot be solved by such one or two people, it is related with the country and government. For instance, there are already some limitations to this kind of car-sharing companies, but stricter laws could be enacted in the future that forbid this possibility and require. So the only thing we should do is to keep this risk away. To avoid this, the team members should do exactly research in society and keep an eye on discussions of these laws, which typically take months to be approved, and be ready to move fast before the legislation is actually enacted.

We also have to consider issues arising from people management inside company. Key members of the team may get ill just prior to important milestones or meetings, or may be ill for long periods of time, causing delays. 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.

Appendix A

Software and tools used

- Microsoft office Word for these kind of paper work.
- GitHub for version control and distributed work.
- Google drive for group organization.
- Excel for drawing tables.

Appendix B

Hours of work

To finish this document, we spent 15 hours per person. We also report here the overall amount of hours per person required by the project.

Document	**Hours of work per person**
Requirements Analysis and Specifications Document (RASD)	30
Design Document (DD)	25
Integration Test Plan Document (ITPD)	20
Project Plan Document (PPD)	5
Total	**80**