POLITECNICO
MILANO 1863

Computer Science and Engineering

PowerEnjoy Service - Project Plan

January 21, 2017

Prof. Luca Mottola

Authors:

- ZHOU YINAN(Mat. 872686)
- ZHAO KAIXIN(Mat. 875464)
- ZHAN YUAN(Mat. 806508)

Contents

1 Introduction

1.1 Revision History

Version 1.0

1.2 Purpose and Scope

This document aims at analyzing the overall complexity and making an estimation about the project size and required effort. The result will help project manager to decide the project budget, resource allocation and the schedule of activities.The document is divided into four parts.

In the first part of the document, We will use two specific methods to estimate the size and complexity of the project. First of all, we will use Function Points to calculate the average line of codes. Secondly, we will use COCOMO method to indicate the cost and effort estimation.

In the second part of the document, we will present the tasks for the project and the corresponding schedule. We will use the above results to come up with a suitable working plan covering the entire project development.

In the third part of the document, we will assign each team member specific missions to tickle down the project.

finally, we are going to analyze the risk we may encounter during the development. By analyzing the risk and coming up with possible solutions, we'll minimize the possibility for failure.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- Precedentedness: High if a product is similar to several previously developed projects
- Development Flexibility: High if there are no specific constraints to conform to pre-established requirements and external interface specs

- Architecture / Risk Resolution: High if we have a good risk management plan, clear definition of budget and schedule, focus on architectural definition
- Team Cohesion: High if all stakeholders are able to work in a team and share the same vision and commitment.
- Process Maturity: Refers to a well known method for assessing the maturity of a software organization, CMM, now evolved into CMMI

1.3.2 Acronyms , Abbreviations

- FP : Function Point
- ILF : Internal Logic File
- ELF : External Logic File
- EI : External Input
- EO : External Output
- EQ : External Inquires
- PREC : Precedentedness
- FLEX : Development Flexibility
- RESL : Risk Resolution
- TEAM : Team Cohesion
- PMAT : Process Maturity

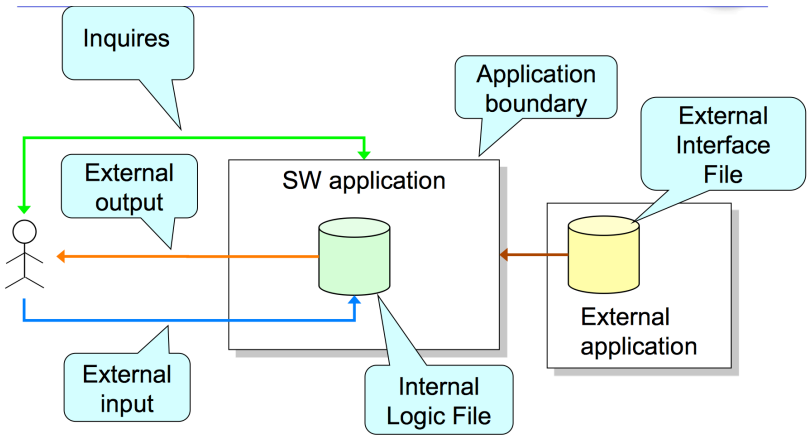
1.4 Reference Documents

- Specification Document Assignments AA 2016-2017
- Function Point tables
- COCOMO tables

2 Project size, cost and effort estimation

We use functional points to estimate the software size. A Function Point (FP) is a unit of measurement to express the amount of business functionality, an information system (as a product) provides to a user. FPs measure software size. They are widely accepted as an industry standard for functional sizing.
Functional Points are based on a combination of program characteristics, more specifically :

- Data structures
- Inputs and outputs
- Inquires
- External interfaces



A weight is associated with each of these FP counts; the total is computed by multiplying each raw count by the weight and summing all partial values. The weight table for different Function types is described below.

	Complexity Weight		
Function Type	*Low*	*Average*	*High*
Internal Logic Files	7	10	15
External Logic Files	5	7	10
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6

2.1 Size estimation: function points

2.1.1 Internal Logic Files(ILFs)

Internal Logical File (ILF): homogeneous set of data used and managed by the application. In our application, there are a few tables we need to manage in the ILFs in order to provide functional requirements.

- User table : The User table maintains all the valid registered users, including the credentials and paymentInfo. More specifically, the credential must include name, email address, login code and number of driving license. Payment information includes number of bank account or credit card, expired time, security code, holder name and Phone number.
- Car table : The table maintains all the information about the cars, including car plate, capacity, current state and onCarDev info.
- Reservation table : The Reservation table maintains all the cars that are currently reserved. The table has a map information between user(email) and car(plate). Also the table maintains the remaining time left for the user to pick up the car.
- Ride table : The Ride table maintains all the cars that are currently under working. The table has a map between user(email) and car(plate).
- DiscountAndPunish table : Whenever a discount or punishment happens, we store the corresponding information in it. The table has the user email, car plate, the discount or punishment amount and a short description.

- Area table : the Area table stores all the safe area information.

Table 1: Internal Logic Files

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

With the Internal Logic Files table above, we can estimate the FP for ILF.

Table 2: ILF

ILF	Complexity	FPs
User	HIGH	15
Car	HIGH	15
Ride	AVG	10
Reservation	AVG	10
DiscountAndPunishment	HIGH	15
TOTAL		65

2.1.2 External Logic Files(ELFs)

The external Logic Files in our application are data source from Google Map and bank services. For the bank services, our system just generate a request for remote bank service and then leave the user to interact with it. So we do not have much to do with the bank services. The Google Map external service, however, requires some modifications about the data we get.

Table 3: External Logic Files

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

Table 4: ELF

ELF	Complexity	FPs
Bank Service	Low	5
Google Map	Avg	7
Total		12

2.1.3 External Inputs(EIs)

Our service deals with several situations which require user inputs.

- Register : Register operations is simple, just needs the client to fill a form and send to the server. The server will then check the validity of the info. For a positive result, a password is generated and send back to the user. The new user info will be inserted into the data base. Thus the complexity for it is low.
- Login : It is simple operation, just needs the server to check the validity of the email and password. The complexity for it is low.

inquires

- Get Available Cars : It is a bit complex operation. The user needs to send to the server the request. The server will need to analyze the location, look for available cars near the location. In order for this operation to proceed, several steps of interaction with different components is required. The complexity is high.
- Reserve : It is a simple operation. The user sends the request to server with the information about the reservation choice. The complexity is low.
- Unlock the Door : It is a simple operation. The user sends the request to the server with the info of his/her location information. The Complexity is low.
- Start Ride : It is a simple operation, only needs to send the request to the server and update the corresponding table. The complexity is low.
- End Ride : Unlike Start Ride, End Ride is a bit more complex. Besides sending the request to the server, the system also needs to compute the total cost, update the corresponding table, and call external bank service. The complexity for this operation is high.

- Modify info : This operation includes modification of existing information in the system. It is a simple operation, since it just needs to check the validity and update the corresponding table. The complexity is low. The average atomic modification includes inserting, updating and deleting. By matching all of them to User, Car and SafeArea. The overall is 3*10 = 30 atomic modifications.

Table 5: External input

	Data Elements		
File Types	1-5	5-15	16+
0-1	Low	Low	Avg
2-3	Low	Avg	High
4+	Avg	High	High

Table 6: EI

EI	Complexity	FPs
Register	Low	3
Log in	Low	3
Get Available Cars	High	6
Reserve	Low	3
Unlock Door	Low	3
Start Ride	Low	3
End Ride	High	6
Modification	Low	3*10
Total		60

2.1.4 External Inquires(EQs)

External quires allow users to retrieve information from the server.

- User retrieve his/her credentials/PaymentInfo : It is a simple operation because it only needs to find info from data base and send back to the user. The complexity is low.
- Get safeAreas : It is a simple operation which only deals with the data base. The complexity is low.
- Get DiscountsAndPunishment : It is a simple operation which only deals with the data base. The complexity is low.

Table 7: External Query

	Data Elements		
File Types	1-5	5-15	16+
0-1	Low	Low	Avg
2-3	Low	Avg	High
4+	Avg	High	High

Table 8: EQ

EQ	Compexity	FPs
Retrive Info	Low	3
Get SafeAreas	Low	3
Get Discounts and Punishment	Low	3
Total		9

2.1.5 External Outputs(EOs)

The system needs to notify the user and the car the corresponding information.

- Notify the user expired Reservation time
- Notify the user completion of the payment.
- Notify the user the execution of punishment.
- Notify the car the current Location and price.

All the operations are fairly simple, except for the last one which needs to interact with several IFLs and EFLs.

Table 9: External Output

	Data Elements		
File Types	1-5	5-15	16+
0-1	Low	Low	Avg
2-3	Low	Avg	High
4+	Avg	High	High

Table 10: EO

EO	Compexity	FPs
Notify expired reservation time	Low	4
Notify the completion of payment	Low	4
Notify the execution of punishment	Low	4
Notify the car Location and Price	High	6
Total		18

2.1.6 Overal estinamtion

The following table summarize the overall estimation.

Table 11: Overall estimation

Function Type	Value
Internal Logic File	65
External Logic File	12
External Input	60
External Query	9
External Output	18
Total	164

Consider using JEE as the development tool, and the additional work for User and Cap Applicationm, we get the following estimated lines of code:

SLOC = 164 * 67 = 12464

76

2.2 Cost and Effort Estimation: COCOMO II

In this section we are going to use the COCOMO II method to estimate the cost and effort which would be needed to development this application – PowerEnjoy.

2.2.1 Scale Drivers

In order to evaluate the cost and effort which should be applied in this project, we refer to the official COCOMO II table which is released:

Table 12: Scale Factor values, SFj, for COCOMO II Models

Scale Factors	Very Low	Low	Normal	High	Very High	Extra High
PREC,SFj	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,SFj	rigorous 5.07	occasional relaxation 4.05	some relaxation 3.04	general conformity 2.03	some conformity 1.01	general goals 0.00
RESL SFj	little (20%) 7.07	some (40%) 5.65	often (60%) 4.24	generally (75%) 2.83	mostly (90%) 1.41	full (100%) 0.00
TEAM,SFj	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 SFj	Level 1 Lower 7.80	Level 1 Upper 6.24	Level 2 4.68	Level 3 3.12	Level 4 1.56	Level 5 0.00

A brief description for each scale driver:

- Precedentedness: Precedentedness would be high if the project is similar to the previous developed projects. So the Precedentedness would be depended on the experience of out team with the development of this kind of project. Since this is the first time for our

team members to manage and develop such a big project, this value should be Low.

- Development Flexibility: Development Flexibility would be high if there are no specific constraints to conform to pre-established requirements and external interface specs. Since in this project, there are strict requirements, but without limitation for the implementation method. This value should be Normal.

- Risk Resolution: Risk Resolution should be high if we have a good risk management plan, clear definition of budget and schedule, focus on architectural definition. As the result of our analysis, we have a great and extensive risk analysis. Therefore, this value should be high.

- Team Cohesion: Team Cohesion should be high if all stakeholders are able to work in a team and share the same vision and commitment. Since the members in our team live in the same city and we know each other perfectly, we can work in a cooperation way. therefore, this value should be very high.

- Process Maturity: Process Maturity refers to a well known method for assessing the maturity of a software organization, CMM, now evolved into CMMI. Although we do not have experience about the development of such a big project, we have achieved all the requirements successfully. And we also have some experience about the Java projects, so this value should be set to Normal.

Overall, the result of our assessment is as follow:

Table 13: Result of Scale Drivers

Scale Driver	Factor	Value
Precedentedness (PREC)	Low	4.96
Development flexibility (FLEX)	Normal	3.04
Risk resolution (RESL)	High	2.83
Team cohesion (TEAM)	Very High	1.10
Process maturity (PMAT)	Normal	4.68
Total		16.61

2.2.2 Cost Drivers

There are 17 Cost Drivers for the Post-Architecture:

- Required Software Reliability (RELY)
- Database size (DATA)
- Product complexity (CPLX)
- Required reusability (RUSE)
- Documentation match to life-cycle needs (DOCU)
- Execution time constraint (TIME)
- Storage constraint (STOR)
- Platform Volatility (PVOL)
- Analyst Capability (ACAP)
- Programmer Capability (PCAP)
- Application Experience (APEX)
- Platform Experience (PLEX)
- Language and Tool Experience (LTEX)
- Personnel continuity (PCON)
- Usage of Software Tools (TOOL)
- Multisite development (SITE)
- Required development schedule (SCED)

We have analysed the Cost Drivers step by step:

- Required Software Reliability (RELY):
  Since the PowerEnjoy is the only way for the user to get the services, this system should be reliable. Otherwise there would be financial loss of the company, and would lead to inconveniences for the users. Therefore, RELY should be High.

Table 14: RELY Cost Drivers

RELY descriptors	slightly inconvenience	easily recoverable losses	moderate recoverable losses	high financial loss	risk to human life	
Rating level	Very low	Low	Normal	High	Very High	Extra High
Effort multipliers	0.82	0.92	1.00	1.10	1.26	n / a

- Database size (DATA):
  This measure considers the effective size of our database. In fact, we have no way to get the extremely precise answer. We can only estimate the Database Size roughly. Since we have estimated the SLOC = 12464, and we set the ratio D/P to be 500. So the DATA Cost Drivers should be High.

Table 15: DATA Cost Drivers

DATA Descriptors		D/P ¡= 10	10 ¡= D/P ¡= 100	100 ¡= D/P ¡= 1000	D/P ¿= 1000	
Rating level	Very Low	Low	Nonimal	High	Very High	Extra High
Effort multipliers	n/a	0.90	1.00	1.14	1.28	n/a

- Product complexity (CPLX):
  Set to High, due to the SLOC is large.

Table 16: CPLX Cost Driver

Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	0.73	0.87	1.00	1.17	1.34	1.74

- Required reusability (RUSE):
  In this project, the codes and documents would only be used by the this project itself. Therefore, the RUSE should be set to Nominal.

Table 17: RUSE Cost Driver

RUSE Descriptors		None	Across project	Across program	Across product line	Across multiple product lines
Rating level	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	0.95	1.00	1.07	1.15	1.24

- Documentation match to life-cycle needs (DOCU):
  This value depends on the relationship between the documents and the requirements. In our project, we have satisfied every requirements for the application. Therefore, this value should be set to High.

Table 18: DOCU Cost Driver

DOCU Descriptors	Many life-cycle needs uncovered	Some life-cycle needs uncovered	Right-sized to life-cycle needs	Excessive for life-cycle needs	Very excessive for life-cycle needs	
Rating level	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	0.81	0.91	1.00	1.11	1.23	n/a

- Execution time constraint (TIME):
  This value depends on the expected usage of CPU when the software is working. Since this application should response rapidly, we suppose the TIME should be set to Very High.

Table 19: TIME Cost Driver

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 level	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	n/a	1.00	1.11	1.29	1.63

- Storage constraint (STOR):
  This value depends on the capability of storage of the hardware when the software is working. Since nowadays the capability of disk drivers can easily reach a high level, and the cost of such kinds of disk drivers would be cheap. Therefore, this value should be set to Nominal.

Table 20: STOR Cost Driver

STOR Descriptors			50% use of available storage	70% use of available storage	85% use of available storage	95% use of available storage
Rating level	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	n/a	1.00	1.05	1.17	1.46

- Platform Volatility (PVOL):
  In fact, we do not expect the version of application changes so often. But the user application may require some new version for satisfy the change of mobile-phone operating system. what's more, some user may want to have some new functions. Therefore, this system may have to be release twice a year. Overall, this value should be set to Nominal.

Table 21: PVOL Cost Driver

PVOL Descriptors		Major change every 12 mo., minor change every 1 mo.	Major: 6mo; minor: 2wk.	Major: 2mo, minor: 1wk	Major: 2wk; minor: 2 days	
Rating level	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	0.87	1.00	1.15	1.30	n/a

- Analyst Capability (ACAP):
  We think we have finished analysis documents appropriately. For this reason, this value should be set to High.

Table 22: ACAP Cost Driver

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

- Programmer Capability (PCAP):
  We have no way to get a extremely precise result for this value, since we would not finish the implementation part. But we can estimate roughly. Since we have finished some Java program, this value should be set to Nominal.

Table 23: PCAP Cost Driver

PCAP Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating level	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.35	1.15	1.00	0.88	0.76	n/a

- Application Experience (APEX):
  We have not experiences about the implementation of J2E project. We only have the experiences about JAVA implementation. Therefore, this value should be set to Low.

Table 24: APEX Cost Driver

APEX Descriptors	¡= 2 months	6 months	1 year	3 year	6 year	
Rating level	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.22	1.10	1.00	0.88	0.81	n/a

- Platform Experience (PLEX):
  We have no experiences about the J2E implementation. But we have experiences about the Database and the Java. Therefore, we set this value to Nominal.

Table 25: PLEX Cost Driver

PLEX Descriptors	¡= 2 months	6 months	1 year	3 year	6 year	
Rating level	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.19	1.09	1.00	0.91	0.85	n/a

- Language and Tool Experience (LTEX):
  We have no experiences about the J2E implementation. But we have experiences about the Database and the Java. Therefore, we set this value to Nominal.

Table 26: LTEX Cost Driver

LTEX Descriptors	¡= 2 months	6 months	1 year	3 year	6 year	
Rating level	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.20	1.09	1.00	0.91	0.84	n/a

- Personnel continuity (PCON):
  Since the time we can spend on this project is quite limited. This value we should set to Very Low.

Table 27: PCON Cost Driver

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

- Usage of Software Tools (TOOL):
  Since we have a very good application implementation environment, we should set this value to High

Table 28: TOOL Cost Driver

TOOL Descriptors	edit, code, debug	simple, frontend, backend CASE, little integration	basic life-cycle tools, moderately integrated	strong, mature life-cycle tools, moderately integrated	strong, mature, proactive life-cycle tools, well integrated with processes, methods, reuse	
Rating level	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.17	1.09	1.00	0.90	0.78	n/a

- Multisite development (SITE):
  The members in our team are live in the same city, and also thanks to the wideband Internet services, we can communicate with each other. Therefore, this value should be set to Very High.

Table 29: SITE Cost Driver

SITE Collocation Descriptors, SITE Communications Descriptors	International, Some phone, mail	Multi-city and multi-company, Individual phone, fax	Multi-city or multi-company, Narrow band email	Same city or metro area,Wideband electronic communication	Same building or complex Wideband elect. comm., occasional video conf.	Fully collocated, Interactive multimedia
Rating level	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.22	1.09	1.00	0.93	0.86	0.80

- Required development schedule (SCED):
  Although our available time in this project is limited, we have worked on this project in a consistent time. Therefore, this value should be Nominal.

Table 30: SCED Cost Driver

SCED Descriptors	75% nominal	85% nominal	100% nominal	130% nominal	160% nominal	
Rating level	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.43	1.14	1.00	1.00	1.00	n/a

Overall, our results are as follows:

Table 31: Result of Cost Drivers

Cost Driver	Factor	Value
Required Software Reliability (RELY)	High	1.10
Database size (DATA)	High	1.14
Product complexity (CPLX)	High	1.17
Required Reusability (RUSE)	Nominal	1.00
Documentation match to life-cycle needs (DOCU)	High	1.11
Execution Time Constraint (TIME)	Very High	1.29
Main storage constraint (STOR)	Nominal	1.00
Platform volatility (PVOL)	Nominal	1.00
Analyst capability (ACAP)	High	0.85
Programmer capability (PCAP)	Nominal	1.00
Application Experience (APEX)	Low	1.10
Platform Experience (PLEX)	Nominal	1.00
Language and Tool Experience (LTEX)	Nominal	1.00
Personnel continuity (PCON)	Very Low	1.29
Usage of Software Tools (TOOL)	High	0.90
Multisite development (SITE)	Very High	0.86
Required development schedule (SCED)	Nominal	1.00
Total		1.96127

2.2.3 Effort Equation

This final equation gives us the effort estimation measured in Person-Months (PM):

Effort = A * EAF * KSLOCE

A = 2.94 (for COCOMO II)
EAF=product of all cost drivers (1.96127)
E = exponent derived from the scale drivers. It is computed as:
B+ 0.01 * SF[i]=B+ 0.01 * 16.61 = 0.91 + 0.1661 =1.0761
in which B is equal to: 0.91 for COCOMO II.

Effort = A * EAF * KSLOCˆE = 2.94 * 1.96127 * 12.464 ˆ1.0761 =
87.081 PM = 87 PM

2.2.4 Schedule Estimation

Regarding the final schedule, we are going to use the following formula:

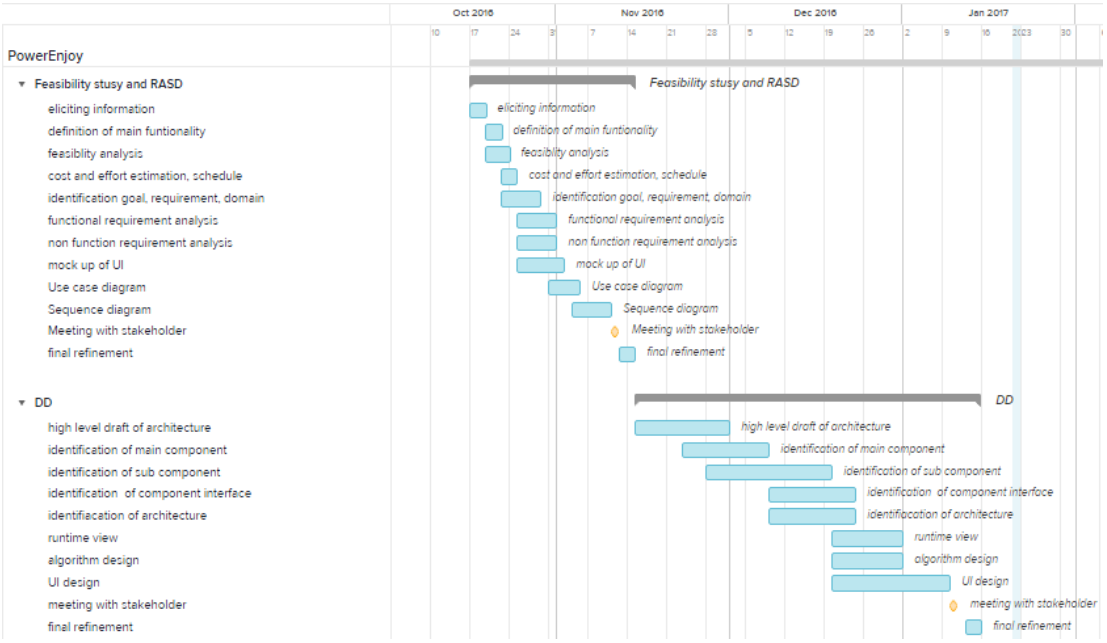
Duration = 3.67 * Effort ^ F

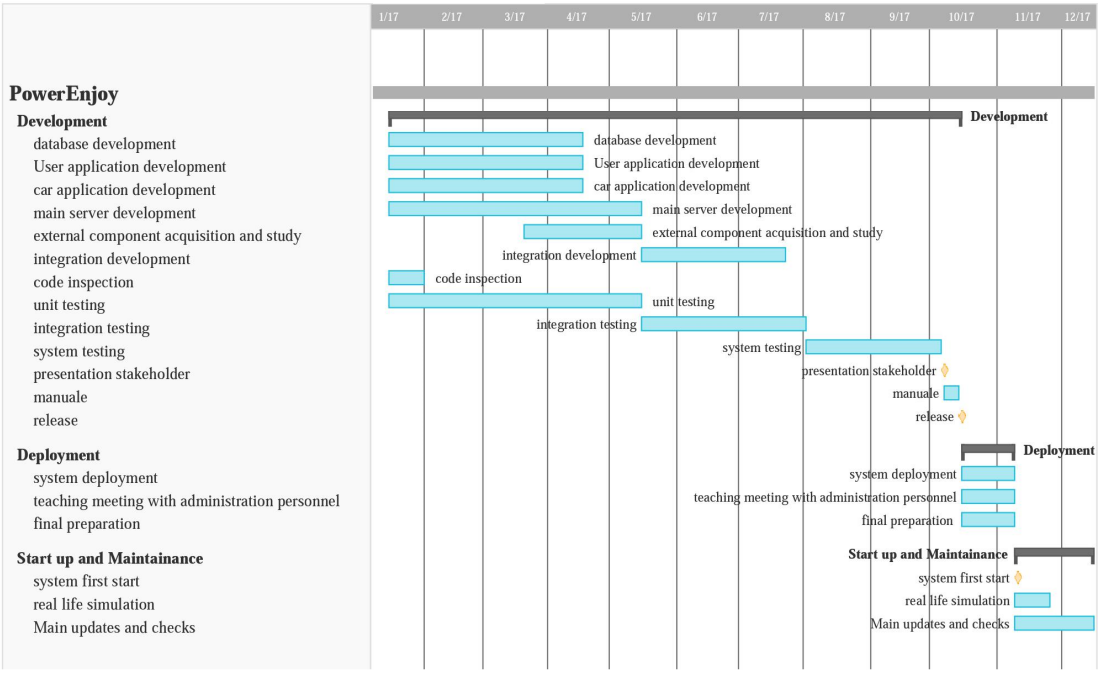
F = 0.28 + 0.2 * (E-B) = 0.28 + 0.2 * (1.0761 - 0.91)= 0.31322
Effort = 87 PM
Duration = 3.67 * 87 ^ 0.31322 = 14.86 months

This is the Schedule which we have estimated.

3 Schedule

In this chapter we are going to provide the general schedule of principal tasks that we will perform to complete our project. The tasks chosen are essential for the completeness of the project. We do not consider the trivial process and the details will be defined during the project.
The entire process follows the phases of waterfall model, therefore dependencies between core activities are sequential. Moreover, to avoid delay caused by tasks waiting for another to complete, we will anticipate the start of the task as much as possible.
For readability, we have split the schedule into two parts, the first one covers the Feasibility study ,RASD and DD. Due to the dependency between activities, the second part starts from the end of DD to the end of the project.
The detail duration estimation will be found in the next chapter.




PowerEnjoy
Development
database development
User application development
car application development
main server development
external component acquisition and study
integration development
code inspection
unit testing
integration testing
system testing
presentation stakeholder
manuale
release
Deployment
system deployment
teaching meeting with administration personnel
final preparation
Start up and Maintainance
system first start
real life simulation
Main updates and checks
1/17
2/17
3/17
4/17
5/17
6/17
7/17
8/17
9/17
10/17
11/17
12/17

4 Resource allocation

In this chapter we going to continue the discourse of how to divide the work between development team. The table below shows the duration and the responsible member of each task.

Aforementioned person will organize detail human resource allocation of each job, aiming to guarantee the completeness within the deadline. So we will not provide the specific staff allocation chart.

PowerEnjoy

Feasibility stusy and RASD	**0%**		**Start**	**Due**	**Assigned**
eliciting information	0%		Oct 17, 2016	Oct 18, 2016	ZHAN, ZHAO, ZHOU
definition of main funtionality	0%		Oct 19, 2016	Oct 20, 2016	ZHAN, ZHAO, ZHOU
feasiblity analysis	0%		Oct 19, 2016	Oct 21, 2016	ZHAN, ZHAO, ZHOU
cost and effort estimation, schedule	0%		Oct 21, 2016	Oct 24, 2016	ZHAN, ZHAO, ZHOU
identification goal, requirement, domain	0%		Oct 21, 2016	Oct 27, 2016	ZHAN, ZHAO, ZHOU
functional requirement analysis	0%		Oct 25, 2016	Oct 31, 2016	ZHOU
non function requirement analysis	0%		Oct 25, 2016	Oct 31, 2016	ZHAO
mock up of UI	0%		Oct 25, 2016	Nov 1, 2016	ZHAN
Use case diagram	0%		Oct 31, 2016	Nov 3, 2016	ZHAN, ZHAO, ZHOU
Sequence diagram	0%		Nov 3, 2016	Nov 9, 2016	ZHAN, ZHAO, ZHOU
Meeting with stakeholder			Nov 10, 2016	Nov 10, 2016	
final refinement	0%		Nov 11, 2016	Nov 14, 2016	ZHAN, ZHAO, ZHOU
DD	**0%**		**Start**	**Due**	**Assigned**
high level draft of architecture	0%		Nov 15, 2016	Nov 30, 2016	ZHAO, ZHOU
identification of main component	0%		Nov 23, 2016	Dec 7, 2016	ZHAO, ZHOU
identification of sub component	0%		Nov 28, 2016	Dec 19, 2016	ZHAN, ZHAO
identification of component interface	0%		Dec 8, 2016	Dec 22, 2016	ZHOU
identifiacation of architecture	0%		Dec 8, 2016	Dec 22, 2016	ZHAO, ZHOU
runtime view	0%		Dec 20, 2016	Dec 30, 2016	ZHAN
algorithm design	0%		Dec 20, 2016	Dec 30, 2016	ZHAN, ZHAO, ZHOU
UI design	0%		Dec 20, 2016	Jan 9, 2017	ZHAN
meeting with stakeholder			Jan 10, 2017	Jan 10, 2017	
final refinement	0%		Jan 12, 2017	Jan 13, 2017	ZHAN, ZHAO, ZHOU

Development	0%		Start	Due	Assigned
database development	0%		Jan 16, 2017	Apr 17, 2017	ZHOU
User application development	0%		Jan 16, 2017	Apr 17, 2017	ZHAO
car application development	0%		Jan 16, 2017	Apr 17, 2017	ZHAN
main server development	0%		Jan 16, 2017	May 15, 2017	ZHAN, ZHAO, ZHOU
external component acquisition and study	0%		Mar 21, 2017	May 15, 2017	ZHAN, ZHAO, ZHOU
integration development	0%		May 16, 2017	Jul 21, 2017	ZHAN, ZHAO, ZHOU
code inspection	0%		Jan 16, 2017	Jan 31, 2017	ZHAN, ZHAO, ZHOU
unit testing	0%		Jan 16, 2017	May 15, 2017	ZHAN, ZHAO, ZHOU
integration testing	0%		May 16, 2017	Aug 1, 2017	ZHAN, ZHAO, ZHOU
system testing	0%		Aug 2, 2017	Oct 4, 2017	ZHAN, ZHAO, ZHOU
presentation stakeholder			Oct 5, 2017	Oct 5, 2017	
manuale	0%		Oct 6, 2017	Oct 12, 2017	ZHAN, ZHAO, ZHOU
Deployment	**0%**		**Start**	**Due**	**Assigned**
system deployment	0%		Oct 16, 2017	Nov 8, 2017	ZHAN, ZHAO, ZHOU
teaching meeting with administration	0%		Oct 16, 2017	Nov 8, 2017	ZHAN, ZHAO, ZHOU
final preparation	0%		Oct 16, 2017	Nov 8, 2017	ZHAN, ZHAO, ZHOU
Start up and Maintainance	**0%**		**Start**	**Due**	**Assigned**
system first start			Nov 9, 2017	Nov 9, 2017	
real life simulation	0%		Nov 9, 2017	Nov 24, 2017	ZHAN, ZHAO, ZHOU
Main updates and checks	0%		Nov 9. 2017	Dec 15. 2017	ZHAN. ZHAO. ZHOU

5 Risk management

In this section, we'll analyze the risk we may encounter during the project development. Basically, there are three kinds of risks : Project risks, Technical risks, and Business risks. We'll present specific risks we may be trapped with and what we can do to deal with it. We use [L,M,H] to indicate the probability the risk will occur. [Low,Moderate,High]

The most possible risks come from the technical part. They threaten the quality and timeliness of the software to be produced. Some possible risks may be :

- Wrong functionality [M]: Wrong functional quality can be a serious risk and lead to meaningless workload which increases the cost and slow down the project.
  To deal with wrong functionality, we adopt a proactive risk strategy. By better writing the RASD document and increase the meeting frequency with the stakeholders, we avoid this kind of risk. Also

frequently report the project process and get feedback from the stakeholders will help.

- Wrong User Interface[M] : Wrong User Interface can lead to meaningless workload.
  To deal with it, we need to frequently meet with the stakeholders and make necessary discussion.

- Bad external components[L] : Bad external components are a major issue and threat to our project. Our project largely depend on the reliability of the following external components : Google Map, DBMS, Bank Service. In case any of them fail, we need to rewrite the business logic components and this leads to significant workload and time increase.

  Although the consequence is serious, the possibility of this risk to happen is actually low. Since the external components we choose to use is from huge and reliable companies like Google and Oracle. So we adopt a reactive risk strategy here.

- Fail to accomplish logic components[L] : This risk is purely technical and there are no good solutions. Either we give programmers time to tick the task down or we recruit new employers with experience. Since the actual problem can be various, it is difficult to come up with a specific prediction. Here we adopt a reactive strategy.

The project risk may arise during the software development. These risks mainly come from the stakeholder side.

- Requirements volatility[M] : Requirement may change during the development. The point is we can not add too much constraints on our stakeholders. Here we adopt a proactive approach by organizing frequently meetings with the stakeholders. By doing this, we may avoid the risk and minimize the negative consequence.

- Unrealistic schedule/budget[M] : This risk comes from both our own side and stakeholder side. During the development we may encounter various uncertain situations and risks which will slow down our process. Also budget may be tight. Here we adopt a reactive approach because we do not know how to come up with a efficient solution until we actually get trapped by the problem.

The third part of the risk is business risk. Some possible situations are presented below:

- Management risk[L] : There is a possibility of losing the support of senior management due to a change in focus or a change in people. Since we assume there are three members in our development team, it is actually kind of critical issue, however the possibility is not high. So we adopt a reactive strategy here.
- Budget risk[L] : Losing budget or personnel commitment is not so likely in our case, so we adopt a reactive strategy here.

6 Effort

- 13/01/2017 ZHOU YINAN 2h document structure and introduction
- 14/012017 ZHOU YINAN 1h Risk management
- 16/01/2017 ZHOU YINAN 2h Function points
- 16/01/2017 ZHAO KAIXIN 2h Scale Drivers
- 17/01/2017 ZHAO KAIXIN 5h Cost Drivers
- 18/01/2017 ZHAO KAIXIN 2h Effort Equations
- 19/01/2017 ZHAN YUAN 5h Schedule graph
- 20/01/2017 ZHAN YUAN 1h Resource