



POLITECNICO
MILANO 1863

Project Plan Document

Version: 1.0

Reda Aissaoui, Jinling Xing, Lidong Zhang

January 22, 2017

Content

1. Introduction
2. Revision History
3. Purpose and Scope
4. Definitions and Abbreviations
5. Reference Documents
6. Project size and effort cost
 1. Size estimation: function points
 2. Data function
 1. ILF: Internal Logical Files
 2. EIF: External Interface Files
 3. Transaction function
 1. EI: External Inputs
 2. EO: External Outputs
 3. EQ: External Inquiries
 4. Overall estimation
 5. Cost and effort estimation: COCOMO II
 6. Software Scale Drivers
 7. Software Cost Drivers
 1. Product
 2. Personal
 3. Platform
 4. Project
 8. Effort equation
 9. Schedule estimation
7. Schedule
8. Resource allocation
9. Risk management
10. Effort spent

1.Introduction

1.1 Revision History

1.2 Purpose and Scope

The main purpose of the Project Plan Document is to analyze the expected complexity of our project and estimate the cost and effort of our project.

Using Function Points and COCOMO approaches, we will give an estimate of the expected size of our project from 2 parts: Software Scale Drivers and Software Cost Drivers. The Software Cost Drivers is divided into 4 parts: product, personal, platform, project.

We will also present a Schedule and Resource Allocation charts. Finally, we will identify the main risk areas that may alter the good execution of the project. We will present some measures in order to mitigate the risks.

1.3 Definitions and Abbreviations

Abbreviations

- ILF: Internal Logical Files
- EIF: External Interface Files
- EI: External Inputs
- EO: External Outputs
- EQ: External Inquiries
- DET: DataElement Types
- FTR: File Types Referenced
- RET: Record Element Types
- UFP: The Unadjusted Function Point
- SLOC: Counting Source Lines of Code
- RELY: Required Software Reliability
- EM: effort multipliers
- PM: Person-Months
- PM/KSLOC: Person-Months/Kilo-Source Lines of Code
- SF(E): Scale Factors
- DATA: Data Base Size
- CPLX: Product Complexity
- RUSE: Required reusability
- DOCU: Documentation match to life-cycle needs
- ACAP: Analyst Capability
- PCAP: Programmer Capability
- PCON: Personnel Continuity
- APEX Application Experience
- PLEX Platform Experience

- LTEX Language and Toolset Experience

1.4 Reference Documents

- Project planning example document
- Project Management Basics
- CII Model Man 2000.0
- Assignments AA 2016-2017

2. Project size, cost and effort estimation

2.1 Size estimation: Function Points

In this section we will present the different functional components: Internal Logical Files (ILF), External Interface Files (EIF), External Inputs (EI), External Outputs (EO) and External Inquiries (EQ). Each one of them is classified by complexity that is assigned depending on its associated file numbers such as Data Element Types (DET), File Types Referenced (FTR) and Record Element Types (RET). We will assign function point to each functional component regarding our project. Afterwards, we will calculate the Unadjusted Function Points (UFP).

ILF/EIF	Data Elements Type		
Record Element Types	1-19	20-50	51+
1	Low	Low	Avg
2-5	Low	Avg	High
6+	Avg	High	High

Table 1: Internal Logical Files (ILF) and External Interface Files (EIF)

EI	Data Elements Type		
Record Element Types	1-4	5-15	16+
0-1	Low	Low	Avg
2	Low	Avg	High
3+	Avg	High	High

Table 2: External Inputs (EI)

EO/EQ	Data Elements Type		
File Type Referenced	1-5	6-19	20+
0-1	Low	Low	Avg
2-3	Low	Avg	High
4+	Avg	High	High

Table 3: External Outputs (EO) and External Inquiries (EQ)

Component	Weight		
	<i>Low</i>	<i>Average</i>	<i>High</i>
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6
Internal Logical Files	7	10	15
External Interface Files	5	7	10

Table 4: Complexity Weight Assignment

2.1.1 Data function

2.1.1.1 ILF: Internal Logical Files

ILFs represent data that is stored and maintained within the boundary of the application. In our system (PowerEnjoy) information is stored through numbers of ILFs.

- **Car data** : As the one of main units, the car, in our system its information has to be stored by system. The information about the car will be saved in a table or joint tables which include color of the car, CarID, car plate number, status(available, reserved, in using or unavailable), state(locked or unlocked), battery level, isCarInSafeArea, etc. For example, a record of the car is inserted into table, for this transaction the **DET** is only 7 and the **RET** is 1 so according to the ILF table we consider the complexity of this part is **Low**.
- **User data** : This contains all the information about users and joint other data, for instance, user accounts, bills information, reservation information and so on. So when a record of user is modified or inserted may associate with different entities in the system. In this case many fields and tables will be saved or changed, thus we consider the complexity of this function point as **Avg**.
- **Reservation data** : In this part, almost all other data are associate with the reservation operation. For example when a reservation is added or deleted or modified all the fields and tables which relate to it will be

changed and saved like the available number of car, the bill table, users information, location and so on. So for this part its complexity of function points we set **High**.

- **Ride data** : For this part, it links to reservation and bill system as well as location information but its data storage is not as much as complex as reservation. When a record of a ride is saved in table the actually saved fields only contain location information. Hence, we consider its complexity as **Low**.
- **Safe areas and charge stations data** : For this part we consider its complexity of function point is **Low** since the operation of this part is fixed and stable even there is a data updating or modifying it will be a small changing.
- **Bill data** : For this part we consider its complexity as **Avg** because the data of this part associate with user data, reservation data, ride data, etc. So when a record of bill is inserted in the table it will link to many attributes.

By using the previously de

finied tables(ILF complexity matrix), this is the count we obtain:

ILF	Complexity	FPS
Car data	Low	7
User data	Average	10
Reservation data	High	15
Ride data	Low	7
Safe areas and charge stations data	Low	7
Bill data	Low	7
Total		53

2.1.1.2 EIF: External Interface Files

EIF is a user identifiable group of logically related data or control information referenced by the application, but maintained within the boundary of another application. The primary intent of an EIF is to hold data referenced through one or more elementary processes within the boundary of the application counted. This means an EIF counted for an application must be in an ILF in another application. In our system(PowerEnjoy) it needs to access two external handlers(APIs):

- **Payment handler** : the process and data storage for this part is simple as we only access the third-party API of payment when a transaction happens, our system only needs to store data and receive data so its complexity is set to **Low**.

- **Google map service** : compare to the Payment handler the Google map API is more frequently used and more complex. For example given a address, get the correspondent coordinates, or return the result of estimation time which means from point to another point when given two locations. So we set this to a **Avg** complexity level.

According to the EIF complexity matrix we get the result as follow:

EIF	Complexity	FPS
Payment	Low	5
Google Map	Average	7
Total		12

2.1.2 Transaction function

2.1.2.1 EI: External Inputs

PowerEnjoy's system requires a multitude of inputs coming from different sources. The first one is the inputs made by the system operator. This includes all the inserts of cars, zones, charging stations and users. The second source is the user as he enters personal information, credentials and reservations. The last source is cars' data that flows from all the fleet. The latter is essential since all the operations are based on the status of cars.

Operator

- Insert cars, zones, charging stations and users: This operations have a low complexity therefore they contribute with 10 FPS all together.
- Validate user: This operation have an average complexity as it requires searching the user then validating his account. This will account for 4 FPS.

User

- User registration: The user need to enter his personal information in order to create an account. In this step, data validation is required. Therefore, this operation have an average complexity. It will represent 4FPS.
- Login: The user enters his credentials and they should be validated at the level of the server. This operation contributes with 4FPS
- Reservations: The user should be able to create new reservation, modify them and also delete them. This is a high complexity operation as it requires to verify the current available cars. The reservation creation will account for 10 FPS while the modification and deletion are 5 FPS each.

Cars

As specified before, this operation need to be performed with high accuracy and timeliness. It involves also the management of different data sources. It is a high complexity operation, so it will account for 10FPs.

EI	Complexity	FPs
Insertions by the operator	Low	3*4
User validation	Average	4
User registration	Average	4
Login	Average	4
Create reservation	High	6
Modify and delete reservation	Low	3*2
Total		36

2.1.2.2 EO: External Outputs

The user needs to communicate with PowerEnjoy system outside the context of an inquiry and also PywerEnjoy system needs to communicate with users. What's occasions they need to communicate with each other, we give thwm as follows:

- Notify the car which ride has been assigned to it
- Notify the user that the reservation has been assigned to a specific car.
- Notify the user the sharing car service.
- Notify the user has been checked in.
- Notify the user has been checked out.
- Notify the bill of the user after ride.
- Notify the car the user who made the reservation has been near the car.
- Notify the car the user has changed his location.

EO	Complexity	FPS
Notification to the car which ride has been assigned to it	Low	4
Notification to the user which car he reserved	Low	4
Notification to the user the sharing car service	Low	4
Notification to the car that the user has been checked in	Low	4
Notification to the car that the user has been checked out	Low	4
Notification to the user the bill he has paid after ride	Low	4
Notification to the car the user who made the reservation has been near the car	Low	4
Notification to the car the user has changed his location	Low	4
Total		32

2.1.2.3 EQ: External Inquiries

An inquiry actually is a data retrieval action, and it is a simple operation with a low complexity. The following provides all the External Inquiries:

- A car can retrieve its complete rides and the bill got by each ride.
- The user can retrieve the history of his reservations and the bill has been paid for the rides associated with the reservations and the economic rides conditions.
- The operator can retrieve the number of cars in a specific zone, the location of current cars in a specific zone and the reservation has been reserved or processed.

$$\text{SLOC} = 160 \times 46 = 7360$$

$$\text{SLOC} = 160 \times 67 = 10720$$

EQ	Complexity	FPS
Retrieve complete rides of cars	Low	3
Retrieve the car's bill got by each ride	Low	3
Retrieve user reservation history	Low	3
Retrieve the user's bill has been paid for the rides	Low	3
Retrieve the user's economic rides	Low	3
Retrieve a list of the number of cars in a specific zone	Low	3
Retrieve the location of current cars in a specific zone	Low	3
Retrieve the reservation has been reserved	Low	3
Retrieve the reservation has been processed	Low	3
Total		27

2.1.3 Overall estimation

The following table summarizes the results of our estimation activity:

Function Type	Value
ILF: Internal Logical Files	53
EIF: External Interface Files	12
EI: External Inputs	36
EO: External Outputs	32
EQ: External Inquiries	27
Total	160

Considering Java Enterprise Edition as a development platform and disregarding the aspects concerning the implementation of the mobile applications, we can estimate the total number of lines of code. Depending on the conversion rate, we have a lower bound of:

and an upper bound of

2.2 Cost and effort estimation: COCOMO II

In order to estimate the size of the project, we will use COCOMO II. According to COCOMO II, We can give the estimate of the expected size of our project regarding 2 parts: Software Scale Drivers and Software Cost Drivers. The Software Cost Drivers is composed of 4 parts: product, personal, platform, project.

2.2.1 Software Scale Drivers

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC SF_j	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 SF_j	rigorous 5.07	occasional relaxation 4.05	some relaxation 3.04	general conformity 2.03	some conformity 1.01	general goals 0.00
RESL SF_j	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 SF_j	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 SF_j	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

Table 5: Scale Factor Values, SF_j , for COCOMO II Models

The table above present some of of the most important factors, defined by COCOMO II, that effect the duration and cost of a project. The scale factors are the following:

- *PREC Precedentness* The experience of working with similar related software system in a large scale. Given that it is our first experience, this factor will be set to low.
- *FLEX Development Flexibility* This scale represents to which extent we should comply to external specifications and specifications. Since the specifications were derived from a broad description of the system, we will set this value to Nominal.
- *RESL Risk Resolution* The amount of risk management that is reserved for the project. As the risk management in our project is average, this will be set to nominal

- *TEAM Team cohesion* It represents the problems that may arise from the project stakeholders. Since this project doesn't involve a big number of stakeholders, we will set this one to Very High
- *PMAT Process Maturity* Defines how much the software engineering process is well established and improved. In our case, we will set this factor to Level 1 as it's our first experience.

From the Scale Factor Value defined by COCOMO II, we obtain the following values for the Scale Factors

Scale Factors	Level	Value
PREC SF_1	Low	4.96
FLEX SF_2	Nominal	3.04
RESL SF_3	Nominal	4.24
TEAM SF_4	Very High	1.10
PMAT SF_5	Level 1	7.80
Total		21.14

Table 6: Estimated scale factors

Afterwards, we need to calculate a scale component using the following formula and $B = 0.91$:

$$E = B + 0.01 * \sum_{j=1}^5 SF_j$$

Applying the numbers we get **E = 1.1214**

2.2.2 Software Cost Drivers

2.2.2.1 Product

- **Required Software Reliability**

If the effect of a software failure is only slight inconvenience then RELY is very low. If a failure would risk human life then RELY is very high. The PowerEnjoy is an essential way to get a car and it can also provide sharing service which is benefit for both of user and car driver, but when the user want to reserve a car to go somewhere, this system still can be replace by other application, like CartoGo and Uber. For this reason, the RELY cost driver is set to nominal.

RELY Cost Driver						
RELY De- scriptors:	slight inconve- nience	low, easily re- coverable losses	moderate, easily re- coverable losses	high fi- nancial loss	risk to hu- man life	
Rating Lev- els	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multi- pliers	0.82	0.92	1.00	1.10	1.26	n/a

- **Data Base Size**

We consider the size of our database. What we need to store are users information, car information, ride information, location information and bill information and something else, so we guess our database will reach a 3GB database. Due to the line of codes will be at least of 10,000 SLOC, the ratio D/P (measured as testing DB bytes/program SLOC) is about 300, which result in the value in this part will be high.

DATA Cost Driver						
DATA* De- scriptors	Testing DB bytes/ Pgm- SLOC <10	10 <= D/P <100	100 <= D/P <1000	D/P >= 1000	90th per- centile	
Rating Lev- els	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multi- pliers	n/a	0.90	1.00	1.14	1.28	n/a

- **Product complexity**

Complexity is divided into five areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations. According to the complexity of our project, we set very high for the CPLX.

CPLX Cost Driver						
Rating Lev- els	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multi- pliers	0.73	0.87	1.00	1.17	1.34	1.74

- **Required reusability**

Since the component's careful design, documentation, and testing has been well down before, so we set the The RUSE cost driver (Required Reusability) as nominal.

RUSE Cost Driver						
RUSE De- scriptors:		none	across project	across program	across product line	across multiple product lines
Rating Lev- els	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multi- pliers	n/a	0.95	1.00	1.07	1.15	1.24

- **Documentation match to life-cycle needs**

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. In our case, the Documents are really detailed and every need of the product life-cycle can be predicted by our Documents, so we set the DOCU as nominal.

DOCU Cost Driver						
DOCU Descriptors:	Many lifecycle needs un- covered	Some lifecycle needs un- covered	Right- sized to life-cycle needs	Excessive for life- cycle needs	Very exces- sive for life-cycle needs	
Rating Lev- els	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multi- pliers	0.81	0.91	1.00	1.11	1.23	n/a

2.2.2.2 Personal

- **Analyst Capability**

The analysis and design ability, efficiency and thoroughness, and the ability to communicate and cooperate are really important. All of the elements have been well done because of our effort. For this reason, this parameter is set to high.

ACAP Cost Driver						
ACAP De- scriptors:	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating Lev- els	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multi- pliers	1.42	1.19	1.00	0.85	0.71	

- **Programmer Capability**

We consider the evaluation of Programmer Capability(PCAP) should be as a team rather than as individuals. The ability, efficiency and thoroughness, and the ability to communicate and cooperate are really important. For this reason, this parameter is set to high.

PCAP Cost Driver						
PCAP De- scriptors:	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating Lev- els	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multi- pliers	1.34	1.15	1.00	0.88	0.76	

- **Personnel Continuity**

In our case, the parameter is nominal. since we spent a lot of time on this project from October 2016 to February 2017 and we still need to make a presentation on March 2017, we have spent half year to do this project. The reason why we can't set this parameter as high or very high is because we only spent our spare time to do this project after our class. For this reason, the PCON is set to nominal.

PCON Cost Driver						
PCON De- scriptors:	48% / year	24% / year	12% / year	6% / year	3% / year	
Rating Lev- els	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multi- pliers	1.29	1.12	1.00	0.90	0.81	

- **Application Experience**

The rating for APEX is dependent on the level of applications experience of the project team developing the software system. We all have more than 4 years study experience of computer science, but our development skills are still limited. For this reason, the PCON is set to nominal.

APEX Cost Driver						
APEX De- scriptors:	<=2 months	6 months	1 year	3 year	6 year	
Rating Lev- els	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multi- pliers	1.22	1.10	1.00	0.88	0.81	n/a

- **Platform Experience**

The usage of platforms, including graphic user interface, database, networking, and distributed middleware capabilities, we have used all the platforms before within a limited time. For this reason, the PLEX is set to nominal.

PLEX Cost Driver						
PLEX De- scriptors:	<=2 months	6 months	1 year	3 year	6 year	
Rating Lev- els	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multi- pliers	1.19	1.09	1.00	0.91	0.85	n/a

- **Language and Toolset Experience**

For our team, the experience on the project's programming language, experience on the project's supporting tool can be set to nominal.

LTEX Cost Driver						
LTEX De- scriptors:	<=2 months	6 months	1 year	3 year	6 year	
Rating Lev- els	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multi- pliers	1.20	1.09	1.00	0.92	0.84	

2.2.2.3 Platform

- **Execution Time Constraint (TIME)**

In order to improve costumers' satisfaction and the stability, fluency of our system we assume that the execution time of the system is short in real scene so a **Nominal** level will be set here.

TIME Cost Driver						
TIME De- scriptors:			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 Lev- els	Very Low	Low	Nominal	High	Very High	Extral High
Effort Multi- pliers	n/a	n/a	1.00	1.11	1.29	1.63

- **Main Storage Constraint (STOR)**

Considering to make the best use of the system resource and the system has enough room to backup important data we will set the rating level as **High**.

STOR Cost Driver						
STOR De- scriptors:			50% use of available storage	70% use of avail- able storage	85% use of avail- able storage	95% use of avail- able storage
Rating Lev- els	Very Low	Low	Nominal	High	Very High	Extral High
Effort Multi- pliers	n/a	n/a	1.00	1.05	1.17	1.46

- **Platform Volatility (PVOL)**

According to the identifier of the PVOL, in our case the platform is the mobile-phone operation system and computer OS. So the platform volatility depends on how often the customer update the OS of their devices. For the stability of our core system we don't expect the platform changes frequently. But as developers We have to update our application periodically to serve customers. Hence the the rating level be set as **Low**.

PVOL Cost Driver						
PVOL De- scriptors:		Major change every 12mo.; Minor change every 1 mo.	Major: 6mo.; Minor:2 wk.	Major: 2mo.; Minor:1 wk.	Major: 2wk.; Minor: 2days	
Rating Lev- els	Very Low	Low	Nominal	High	Very High	Extral High
Effort Multi- pliers	n/a	0.87	1.00	1.15	1.30	n/a

2.2.2.4 Project

- **Use of Software Tools (TOOL)**

Our project environment is quite complete and it should be strong and a tool with mature life cycle and moderately integrated due to this the parameter will be set as **High**.

TOOL Cost Driver						
TOOL De- scriptors	edit, code, debug	simple, front end, backend- CASE, little inte- gration	basic lifecycle tools, moder- ately inte- grated	strong, mature lifecycle tools, moder- ately inte- grated	strong, mature, proactive life-cycle tools, well in- tegrated with pro- cesses, methods, reuse	
Rating Lev- els	Very Low	Low	Nominal	High	Very High	Extral High
Effort Multi- pliers	1.17	1.09	1.00	0.90	0.78	n/a

- **Multisite Development (SITE)**

Since our team have a project meeting every week and we live in same city, we communicate with each other by social media software like whats APP and Facebook when we are not working together so we set the rating level of this part as **High**.

SITE Cost Driver						
SITE: Col- location Descriptors:	International	Multi- city and Multi company	Multi- city or Multi company	Same city or metro.area	Same build- ing or complex	Fully col- located
SITE: Com- munications Descriptors:	Some phone, mail	Individual phone, FAX	Narrow band email	Wide band electronic communi- cation.	Wide band elect. comm., occa- sional video conf.	Interactive multime- dia
Rating Lev- els	Very Low	Low	Nominal	High	Very High	Extral High
Effort Multi- pliers	1.22	1.09	1.00	0.93	0.86	0.80

- **Required Development Schedule (SCED)**

We will set this parameter as **Nominal** which the value is 1.00. As a programmer it is our duty to finish our job and present a completed project to customer on time.

SCED Cost Driver						
SCED De- scriptors	75%of nominal	85%of nominal	100%of nominal	130%of nominal	160%of nominal	
Rating Lev- els	Very Low	Low	Nominal	High	Very High	Extral High
Effort Multi- pliers	1.43	1.14	1.00	1.00	1.00	n/a

Overall, our results are summarized by the following table:

<i>Cost Driver</i>	<i>Factor</i>	<i>Value</i>
Required Software Reliability (RELY)	Nominal	1.00
Database size (DATA)	High	1.14
Product complexity (CPLX)	Very high	1.34
Required Reusability (RUSE)	Nominal	1.00
Documentation match to life-cycle needs(DOCU)	Nominal	1.00
Analyst capability (ACAP)	High	0.85
Programmer capability (PCAP)	High	0.88
Personnel Continuity (PCON)	Nominal	1.00
Application Experience (APEX)	Nominal	1.00
Platform Experience(PLEX)	Nominal	1.00
Language and Tool Experience (LTEX)	Nominal	1.00
Execution Time Constraint (TIME)	Nominal	1.00
Main storage constraint (STOR)	High	1.05
Platform volatility (PVOL)	Low	0.87
Usage of Software Tools (TOOL)	High	0.90
Multisite development (SITE)	High	0.93
Required development schedule (SCED)	Nominal	1.00
Product of all cost drivers		0.87367

2.2.3 Effort equation

According to COCOMO II manuel we got Eq.1 to calculate the value of Effort:

$$Effort = A \cdot EAF \cdot KSLOC^E \text{ (Eq.1)}$$

where $A = 2.94$ (from COCOMO II), $EAF = 0.87367$ (calculated from cost drivers analysis), $E = 1.1214$ (calculated from scale drivers analysis), and KSLOC was calculated before in function points section and we got 7.360 as the lower bound of KSLOC and 10.720 as its upper bound.

So the final result as following:

- $Effort_{Lower} = 24.0886$
- $Effort_{Upper} = 36.7244$

2.2.4 Schedule estimation

The formula of duration as following:

$$Duration = 3.67 \cdot Effort^F$$

where $F = 0.28 + 0.2(E - B)$ and $B = 0.91$ according to COCOMO II manual. Since the value of Effort is in a range so for the upper bound of duration we got:

- $Duration_{Upper} = 11.7225$ months

The lower bound of duration :

- $Duration_{Lower} = 10.2328$ months

From the results, the overall time of the entire project is around 10 month but no more than 12 month. And the result is quite close to our estimated time.

3.Schedule

It is necessary to set a schedule to make sure the project is proceeded as planned and finished on time. So we created a project schedule by using Gantt Project tool to keep the project goes well as we expected. In this schedule will show when the project is started and what we did during each period. We also assume that the project will be continued to maintain the integrality and authenticity of the entire development process. So we added the 'Implementation and support' and 'Testing' as well as 'Evaluation' procedures though these will not be performed at the end of the project.

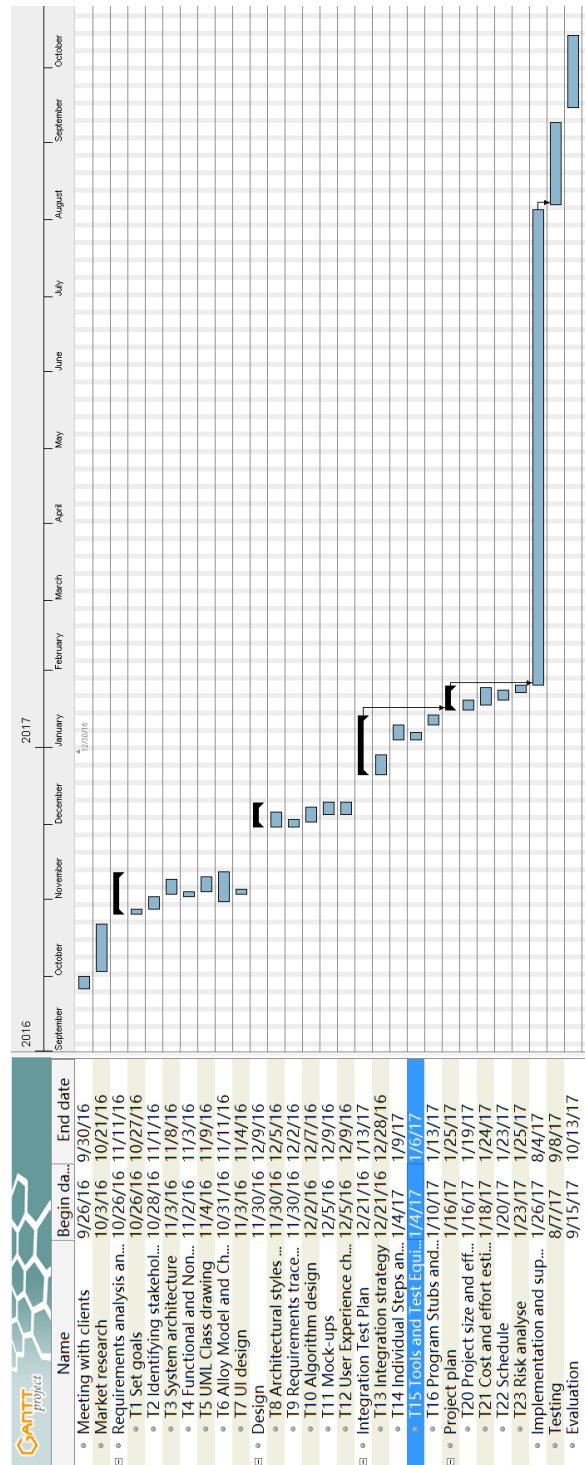


Figure 1: Project's Gantt Chart

4.Resource allocation

In this section we will present how we distribute our team resource to finish our project by creating a Gantt chart. In the Gantt Chart, it can be shown that each of us is responsible for each part of the project and how much time we spent on each of tasks. Note that every time there will be a group meeting before we starting our single task. So that we can avoid misunderstanding and conflict among team members when we are working separately. Again we also assume that we are working at developing phase so in the chart there are incomplete tasks.

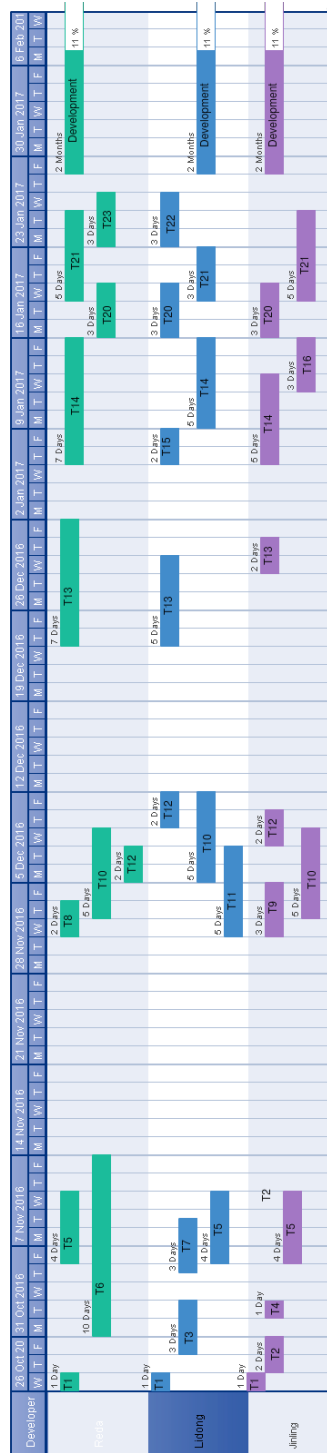


Figure 2: Resources Allocation Diagram

5. Risk management

The following section serves as a basis for risk management. We will identify the main risks that may arise during the execution of our project's schedule. The main areas assessed in risk identification are new technologies, user and functional requirements, system architecture, performance and finally organizational. In order to monitor and mitigate risk, we will present as well some precautions and good practices to achieve our objective.

The first area is **New and Unproven Technologies**. The main business objective is the management of a fleet of "smart" electric cars that connect to PowerEnjoy servers through internet. We will have to gather car data in timely and real-time manner. Difficulties may arise when trying to keep an up-to-date data store of the situation of the cars. Data, for example, may be out of date and users are shown expired information. Consultation with engineers that took part in similar projects is advised to identify the main difficulties. Car communication is the new technology in this project. The server side, since it will be mainly developed in JAVA which is a proven technology, will not present a high risk.

User and Functional Requirements may represent a high risk that will push us to steer into other directions. This is mainly because the user and functional requirements gathering is a tedious process. Furthermore, the requirements may change or other ones may arise during discovery and integrations process. Therefore, the requirements should be well-defined and validated with all the stakeholders of the project. In addition, a sufficient margin should be planned in the schedule to allow some time to adapt to new requirements changes. This margin is also useful if another risk caused the slowing down of the development process.

Another area that should be thoroughly inspected is the **Application and System Architecture**. The decisions made in this area represent the backbone of the application. They guide the design and other important decisions later on in the process. A wrong decision may cause delays if inadequate architecture decisions are made. In this case, there will be a need of a design refactoring and implementation changes. Research about experience in similar application is needed to reduce the risk coming from this area.

Moreover **Performance** should be assessed for all the duration of the project. Performance benchmarks should be performed at each step to avoid discovering performance problems at the end of the project. Key performance indicators should be well-defined and checked. This project relies a lot on real-time data, so performance should be closely monitored.

Last but not least, **Organizational** risks have to be assessed as well. Project management should plan for an efficient execution of the project while keeping a suitable balance between the resources of the project and the expectation of the client. Reports and documentation should be produced at each milestone of

the project. Through the communication and validation of these documentation with the stakeholders, we are sure that we are in a good track while recognizing problems before they cause significant delay.

Code management, centralization and back-up should be performed. Version control is essential to keep back-ups, history and code sharing among the different developers. It is also a useful tool to keep track of how much work is being done and by whom.

Stakeholders are also important in our risk analysis. Since the project is to be deployed in a city, many stakeholders are taking part of it. City council and project sponsors have an important role in the decision making. Therefore, they should be always kept in contact as stated before. The legal environment of the project should as well be kept in consideration. There should be a strong legal consultation about the regulations.

To sum up, many risks may arise during the project execution. Although some of the risks are presented above, we should keep in mind that other unapparent risks may arise. A sufficient time margin should be included in the schedule in order to meet the project deadlines.

6.Effort spent

Lidong Zhang

- 16/01/2017 2h
- 18/01/2017 4h
- 19/01/2017 2h
- 20/01/2017 4h
- 21/10/2017 3h

Jinling Xing

- 17/01/2017 2h
- 19/01/2017 6.5h
- 21/01/2017 5h

Reda Aissaoui

- 16/01/2017 1h
- 18/01/2017 3h
- 19/01/2017 4h
- 21/01/2017 2h
- 22/01/2017 1h