POLITECNICO
MILANO 1863

Project Plan Document

Version: 1.0

Reda Aissaoui, Jinling Xing, Lidong Zhang

January 21, 2017

Content

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.

By means of the Function Points and COCOMO approaches, we can give the estimate of the expected size of our project from 2 parts: Software Scale Drivers and Software Cost Drivers. The Software Cost Drivers divides into 4 parts: product, personal, platform, project.

Finally, we're going to work on Schedule and Resource allocation and evaluation on the possible risks that PowerEnjoy could face.

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_modelman2000.0
- Assignments AA 2016-2017
- Project Management Basics + Advanced Dec. 1

2.Project size, cost and effort estimation

2.1 Size estimation: Function Points

Complexity matrix for function points

Internal Logical Files (ILF), External Interface Files (EIF), External Inputs (EI), External Outputs (EO) and External Inquiries (EQ). Each functional component is classified as a certain complexity based on its associated file numbers such as Data Element Types (DET), File Types Referenced (FTR) and Record Element Types (RET). The complexity matrix for the five components is shown in Table 1. Table 2 illustrates how each function component is then assigned a weight according to its complexity. The Unadjusted Function Point (UFP) is calculated with Equation 1, where Wij are the complexity weights and Zij are the counts for each function component.

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)

	Weight		
Component	*Low*	*Average*	*High*
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 infromation 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 complax 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

fined 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 three 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 opint 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 PywerEnjoy 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 accutally is a data retrieval action, and it is a simple operation with a low complexitY. The follows 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.

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

2.2 Cost and effort estimation: COCOMO II

we can give the estimate of the expected size of our project from 2 parts: Software Scale Drivers and Software Cost Drivers. The Software Cost Drivers divides into 4 parts: product, personal, platform, project.

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC SF_j	thoroughly unpreceden ted 6.20	largely unpreceden ted 4.96	somewhat unpreceden ted 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 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 its 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 Descriptors:	slight inconvenience	low, easily recoverable losses	moderate, easily recoverable 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

- Data Base Size

We consider the size of our database.What wo 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 lease 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* Descriptors	Testing DB bytes/ Pgm-SLOC <10	10 <= D/P <100	100 <= D/P <1000	D/P >= 1000	90th percentile	
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

- 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.

Component Complexity Ratings Levels					
	Control Operations	Computational Operations	Device dependent Operations	Data Management Operations	User Interface Management Operations
Very Low	Straight-line code with a few non-nested structured programming operators: DOs,CASEs, IFTHEN-ELSEs. Simple module composition via procedure callsor simplescripts.	Evaluation of simple expressions: e.g., A=B+C*(DE)	Simple read, write statements with simple formats.	Simple arrays in main memory. Simple COTSDB queries,updates.	Simple inputforms, report generators.
Low	Straight forward nesting of structured programming operators. Mostly simplep redicates	Evaluation of moderate-level expressions: e.g., D=SQRT(B**2-4.*A*C)	No cognizance needed of particular processor or I/O device characteristics I/O done at GET/PUT level.	Single file subsetting with no datas tructure changes, noedits, no intermediate files. Moderately complex COTSDB queries, updates.	Use of simple graphic user interface (GUI) builders.
Nominal	Mostly simple nesting. Some inter module control. Decision tables. Simple call backs or message passing, including middleware supported distributed processing	Use of standard math and statistical routines. Basic matrix/vector operations.	I/O processing includes device selection, status checking and error processing.	Multi-file input and single file output. Simple structural changes, simple edits. Complex COTS-DB queries, updates.	Simple use of widget set.
High	Highly nested structured programming operators with many compound predicates. Queue and stack control.Homogeneous, distributed processing. Single process or soft real-time control.	Basic numerical analysis: multivariate interpolation, ordinary differential equations. Basic truncation, round-off concerns.	Operations at physical I/O level (physical storage address translations; seeks, reads,etc.). Optimized I/O overlap.	Simple triggers activated by data stream contents. Complex data restructuring.	Widget set development and extension. Simple voice I/O, multimedia.
Very High	Reentrant and recursive coding. Fixed priority interrupt handling. Task synchronization, complex callbacks, heterogeneous distributed processing. Single processor hard real-time control.	Difficult but structured numerical analysis: near singular matrix equations, partial differential equations. Simple parallelization.	Routines for interrupt diagnosis, servicing, masking. Communication line handling. Performance intensive embedded systems.	Distributed database coordination. Complex triggers. Search optimization.	Moderately complex 2D/3D, dynamic graphics, multimedia.
Extra High	Multiple resource scheduling with dynamically changing priorities. Microcode-level control.Distributed hard real-time control.	Difficult and unstructured numericalanalysis: highly accurate analysis of noisy, stochastic data. Complex parallelization	Device timing dependent coding, micro programmed operations. Performance critical embedded systems.	Highly coupled, dynamic relational and object structures. Natural language data management.	Complex multimedia, virtual reality, natural language interface.

CPLX Cost Driver						
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

- 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 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

- 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 uncovered	Some lifecycle needs uncovered	Right-sized 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

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 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	

- 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 Descriptors:	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.34	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 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	

- 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 Descriptors:	<=2 months	6 months	1 year	3 year	6 year	
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

- 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 Descriptors:	<=2 months	6 months	1 year	3 year	6 year	
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

- 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 Descriptors:	<=2 months	6 months	1 year	3 year	6 year	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	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.

- **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**.

- **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**.

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 wil be set as **High**.

- **Multisite Development (SITE)**

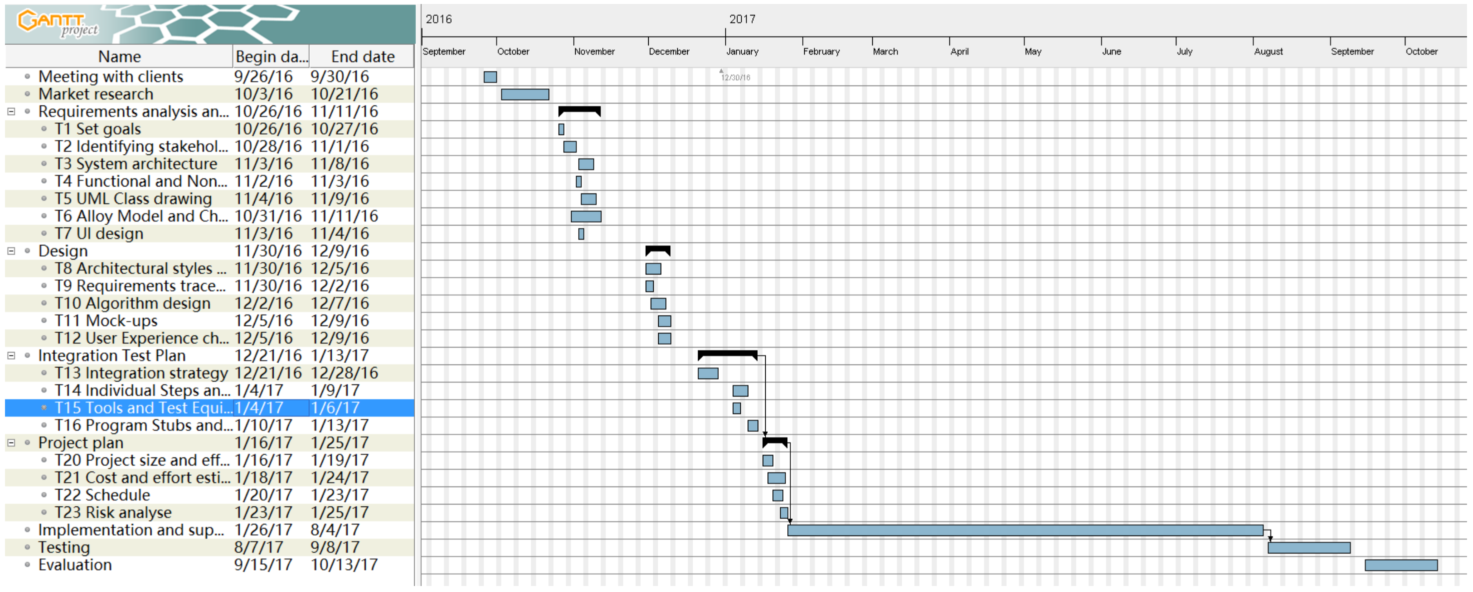
Since our team have a project meeting every week and we live in same city, we communicate with each other by social media software like whats APP and Facebook when we are not working together so we set the rating level of this part as **High**.

- **Required Development Schedule (SCED)**

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

3.Schedule

It is necessary to set a schedule to make sure the project is proceeded as plan and finished on time. So we created a project schedule by using GanttProject to keep the project goes well as we expected. In this schedule will show how we started this project and what we did during each period. We also assume that the project will be continued to maintian 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.



Name	Begin da...	End date
Meeting with clients	9/26/16	9/30/16
Market research	10/3/16	10/21/16
Requirements analysis an...	10/26/16	11/11/16
T1 Set goals	10/26/16	10/27/16
T2 Identifying stakehol...	10/28/16	11/1/16
T3 System architecture	11/3/16	11/8/16
T4 Functional and Non...	11/2/16	11/3/16
T5 UML Class drawing	11/4/16	11/9/16
T6 Alloy Model and Ch...	10/31/16	11/11/16
T7 UI design	11/3/16	11/4/16
Design	11/30/16	12/9/16
T8 Architectural styles ...	11/30/16	12/5/16
T9 Requirements trace...	11/30/16	12/2/16
T10 Algorithm design	12/2/16	12/7/16
T11 Mock-ups	12/5/16	12/9/16
T12 User Experience ch...	12/5/16	12/9/16
Integration Test Plan	12/21/16	1/13/17
T13 Integration strategy	12/21/16	12/28/16
T14 Individual Steps an...	1/4/17	1/9/17
T15 Tools and Test Equi...	1/4/17	1/6/17
T16 Program Stubs and...	1/10/17	1/13/17
Project plan	1/16/17	1/25/17
T20 Project size and eff...	1/16/17	1/19/17
T21 Cost and effort esti...	1/18/17	1/24/17
T22 Schedule	1/20/17	1/23/17
T23 Risk analyse	1/23/17	1/25/17
Implementation and sup...	1/26/17	8/4/17
Testing	8/7/17	9/8/17
Evaluation	9/15/17	10/13/17

Figure 1: Project's Gantter Chart

4.Resource allocation

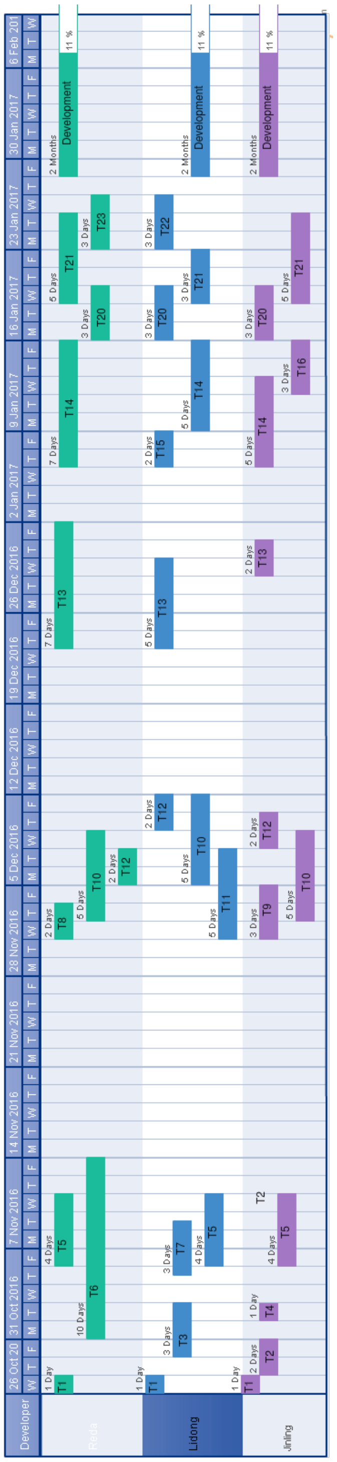


Figure 2: Resources Allocation Diagram

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