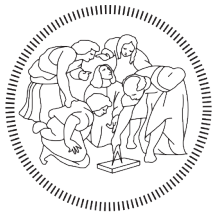

POLITECNICO
MILANO 1863

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.

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

1.4 Reference Documents

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** :

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** :

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:

Table 5: My caption

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 6: 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 Precedenteness* 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 pro-

cess 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 7: 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 $\mathbf{E = 1.1214}$

2.2.1 Software Scale Drivers

2.2.2 Software Cost Drivers

2.2.2.1 Product

2.2.2.2 Personal

2.2.2.3 Platform

2.2.2.4 Project

3.Schedule

4.Resource allocation

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 **Perfomance** 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