Politecnico di Milano
MSc Computer Science and Engineering

Software Engineering 2
Academic Year 2016-2017

Project Plan Document

PowerEnJoy

Authors:

Melloni Giulio 876279
Renzi Marco 878269
Testa Filippo 875456

Reference Professor:
Mottola Luca

Release Date: January 22th, 2017
Version 1.0

Table of Contents

List of Figures

1 | Introduction

1.1 Revision History

Version	Date	Summary
1.0	22/01/2017	First release of the document

1.2 Purpose and Scope

The Project Plan Document (later referred with the acronym *PPD*) refers to the issues that concern the development process of the system. In particular, it covers the critical phase of the estimation of the project in terms of size, cost and time effort. In order to carry out this type of analysis, different methodologies will be used as explained in the second chapter; the result obtained consider a quite large set of various aspects, each one properly represented by a numerical factor. It's important to notice that the assignment of a specific value to these factors is not trivial at all.
The second key point of this document is about the Schedule and Resource Allocation. After having identified all the tasks that have to be executed to complete the project, it is important to define when these different activities should be done. Going down into details, in the Schedule section, it is defined a temporal plan which visually explain the date and time organization of the tasks to be carried out; this then infer a sorting between the tasks. On the other hand, the Resource Allocation establishes for each member of the team the tasks that he to do in the different phases of the project; moreover, while some of them are resource consuming (thus they will be executed from more than 1 person at the same time) , others are strictly sequential so it impossible to be executed simultaneously. At this point is quite clear that there is a strong correlation between the Schedule and Resource Allocation part.
Lastly, it's fundamental to study all the risks that can arise through the different phases of the project development (such as requirements analysis, design of the system architecture, code implementation and testing, deployment and start up) but also while the project is completed and the developed system is running. The identified risks can be grouped into a few groups by considering the aspects they affect the most and for every single task there is an estimation of the impact that it can have on the system and a strategy on how to deal with it.

1.3 Definitions, Acronyms, Abbreviations

PowerEnJoy is the name of the system that has to be developed.

System sometimes called also *system-to-be*, represents the application that will be described and implemented. In particular, its structure and implementation will be explained in the following documents. People that will use the car-sharing service will interact with it, via some interfaces, in order to complete some operations (e.g.: reservation and renting).

Renting it is the act of picking-up an available car and of starting to drive.

Ride the event of picking-up a car, driving through the city and parking it. Every Ride is associated to a single user and to a single car.

Reservation it is the action of booking an available car.

Car a car is an electrical vehicle that will be used by a registered user.

Not Registered User indicates a person who hasn't registered to the system yet; for this reason he can't access to any of the offered function. The only possible action that he can carry out is the registration to get a personal account.

Registered User interacts with the system to use the sharing service. He has an account (which contains personal information, driving license number and payment data) that must be used to access to the application in order to exploit all the functionalities.

Employee it's a person who works for the company, whose main task is to plug into the power grid those cars that haven't been plugged in by the users. He is also in charge of taking care of the status of the cars and of moving the vehicles from a safe area to a charging area and vice versa if needed.

Safe Area indicates a set of parking lots where the users have to leave the car at the end of the rent; the set of the Safe Areas is pre-defined by the system management. These areas are spread all over the city.

Plug defines the electrical component that physically connects the car to the power grid.

Charging Area is a special *Safe Area* that also provides a certain number of plugs that connect the cars to the power grid in order to recharge the battery.

Registration the procedure that an unregistered user has to perform to become a registered user. At the end, the unregistered user will have an account. To complete this operation three different types of data are required: personal information, driving license number and payment info.

Search this functionality lets the registered user search for available cars within a certain range from his/her current position or from a specified address.

RASD is the acronym of *Requirements Analysis and Specification Document*

DD is the acronym of *Design Document*

ITPD is the acronym of *Integration Test Plan Document*

PPD is the acronym of *Project Plan Document*

1.4 Reference Documents

- Project Assignments 2016-2017
- RASD v1.1
- DD v1.1
- ITPD v1.0

2 | Project size, cost and effort estimation

This section talks about the estimation of the size of the project, giving an estimation of lines of code that must be written. This evaluation comes from the calculation of the lines of code that occur to write both the Business Logic part and the User Interface part.
Function Point is the approach used to simulate how the project will be expensive in terms of lines of code.
From this estimation it will be easy to find how much time will be spent on developing the Business Logic part of the project.

2.1 Size Estimation: Function Points

Functionalities that are to be developed and their complexity are provided by the Function Point approach.
In each table is reported how much load can take developing these parts: from *Low*, not so much expensive, to *High*, the heavier.

Internal Logic Files & External Logic Files

		Data Element		
Record Elements		1-19	20-49	50+
	1	Low	Low	Adw
	2-5	Low	Adw	High
	6+	Adw	High	High

Each line of this table underlines how it becomes more heavy the computation with the increasing of *Data Elements* and *Record Elements*.
Adding few records or some data, the program is rapidly loading, stretching the response time.

External Outputs & External Inputs

		Data Element		
Record Elements		1-19	20-49	50+
	1	Low	Low	Adw
	2-5	Low	Adw	High
	6+	Adw	High	High

With *External Outputs* and *External Inputs*, the size of Data and Records must be smaller than the previous ones seen in *Internal Logic Files* and *External Logic Files*.
There is a big load adding only one record, or a few data, getting worst the performance of the system.

External Inquiries

		Data Element		
Record Elements		1-19	20-49	50+
	1	Low	Low	Adw
	2-5	Low	Adw	High
	6+	Adw	High	High

The parts of External Input is the heavier, because it is an interaction between the User and the System.
The table is so heavier at littler variations. One record, or few data are enough to increase the performance of the system.

Complexity Weights

		Complexity Weight		
Function Type	Acronym	*Low*	*Mid*	*High*
Internal Logic Files	ILFs	7	9	15
External Logic Files	ELFs	5	7	10
External Inputs	EIs	2	4	6
External Inquiries	EQs	6	8	10
External Outputs	EOs	2	3	4

The general complexity is so defined.
It is obviously heavier a Function Type like the *Internal Logic Files* or an *External Inquiries*, which manage the update of the Data and the communication between the User and the System, than a *External Output*, *External Input* or an *External Logic Files*, that gives only an output or take an input.

2.1.1 Internal Logic Files: ILFs

Some functionalities store information into the PowerEnJoy system, in order to remember data and let the system use and manage them.
The login is one of these functionalities: the system has to store the *User Data* in order to recognize him. When the data is stored, it will be composed by many fields explaining the personal information, in order to characterize every person.
Safe Area, *Charging Area* and *Car Data* are inserted by the system at each time a new station or vehicle is added into the system. *Safe Area* and *Charging Area* will be condensed in the same table, but diversified by some attributes, in order to have an increase of performance, searching for the data in only one place.

Differently, *Reservation* is continuously added, through a new operation on the app.

ILFs	*Complexity*	*Points*
User Data	Mid	9
Employee Data	Low	7
Safe Area Data	Low	7
Charging Area Data	Low	7
Reservation Data	Mid	9
Ride Data	Mid	9
Car Data	Low	7
Total		55

User Data, *Reservation Data* and *Ride Data* have a *mid* complexity because they have more attributes that will be managed. Differently, *Employee Data*, *Safe Area Data* and *Car Data* have a *Low* complexity because they have few attributes.

2.1.2 External Logic Files: ELFs

The application manages the interaction and the system acquires information from the external services used. *Payment Data* and *Map Data* are considered External Logic Files, because they are provided by external component.
Because of these components are not part of the system, PowerEnJoy will obtain information thanks to external APIs in order to link internal to external data.
Payment Data will be more complex to develop, because the system has to guarantee the security of the payment and of the information, avoiding every type of external attack.

ELFs	*Complexity*	*Points*
Maps Data	Mid	7
Payment Data	High	10
Total		17

In this table, *Payment Data* part is considered more complex than the *Map Data* one, because it has to guarantee security, despite the performance.

2.1.3 External Inputs: EIs

This section underlines the methods called by an operation of the User on the application. The action is so called a client-server action, because the system reacts to an User operation. There are two different type of these changes:

- **UI Action**: the user click some buttons, or insert some text on the User Interface of the application, or on the car screen.
- **Physical Action**: the user plugs the car in a Charging Area.

The most complex will be the creation of the reservation, because it has to interact with data, creating them and inserting them linking the User Interface to the remote DBMS.
It is easy to see how is more easy to develop the part that check if the car has been plugged or not, after the stop of the ride.

EIs	*Complexity*	*Points*
Login	Mid	4
Logout	Low	2
Registration	Mid	4
Insert Reservation	High	6
Delete Reservation	Mid	4
Car Plug In	High	6
Update Car Status	Low	2
Insert a new car	Mid	4
Update a car	Mid	4
Delete a car	Mid	4
Insert a new charging area	High	6
Update charging area	Mid	4
Delete charging area	Mid	4
Convey end ride via touch screen	Mid	4
Insert final destination	Mid	4
Total		62

Insert Reservation, *Insert a new charging area* are considered *High* as complexity, because they interact directly and manually to the *DBMS*. For the same reason, *Delete Reservation*, *Delete a car*, *Update a charging area* and *Delete a charging area* are consider *Mid* complex because.
Car Plug In has an *High* complexity because the car has to interact with the system.
Login, *Logout*, *Registration*, *Update Car Status*, *Convey end ride* and *Insert Final Destination* are all considered as *Mid* or *Low* complexity because they manage few data or the communication is created between Mobile Device - System.

2.1.4 External Inquiries: EQs

These operations that involves an input and an output are the *Ride* actions: when it starts and when it stops.
Starting the ride, the user interact with the Car UI so that the Car System can communicate to the Server that the Ride is started, and the Server can change User Interface on board displaying his navigation.
Otherwise, the system will calculate the cost, when the user interacts with the UI and it will display temporary how much he has to pay.
So *Ride Start* and *Ride Stop* are easily considered as high complex to develop, because they have different communication between the car and the system.
Differently, the *Search* for a car or the *Pick Up* of it, are actions which involve only one communication car - system, so they are considered easier to develop.

EQs	*Complexity*	*Points*
Search for available cars	Mid	8
Show cars information	Mid	8
Enable MoneySavingOption	High	10
Pick Up a car	High	10
Ride initialization	HIgh	8
Total		44

Pick Up a Car, *Ride initialization* and *Money Saving Option* are considered all with an *high* complexity, because they have to establish a communication between Car and System. Differently, the *Search for available cars* and the *Show cars informations* operations are considered *Mid*, because it is easier to link Mobile App to the System.

2.1.5 External Outputs: EOs

External Outputs are elementary operations that generates data that are presented to the User, without any client action.
The cost notification is one example from EOs section: the user, stopping the ride, gets out of the car, and after a certain time the system checks if he has plugged or not his vehicle. After this, the system displays how much the user has to pay.
This is an example of an operation send by the system without any actions of the user.

EOs	*Complexity*	*Points*
Payment Notification	Low	2
Notify Expired Reservation	Mid	4
Total		6

Payment Notification is considered less complex than the *Notify Expired Reservation* because it has to check less fields and to screen a simple interface with the cost.

2.1.6 Overall Estimation

Considering the whole estimation of the Function Types, the total will be the summation of each one of the Total previously found.

Function Type	*Acronym*	*Value*
Internal Logic Files	ILFs	55
External Logic Files	ELFs	17
External Inputs	EIs	62
External Inquiries	EQs	44
External Outputs	EOs	6
Total		184

Having this result, the total lines of code are easily calculable using *conversion factor* and the *usage percentage* used to specify how much the project is divided in different languages. Using this method we can consider the entire project, from the Business Logic to the UI development part, adopting a weighted average between languages:

Language	**Conversion factor (SLOC/FP)**	**Usage percentage**
HTML	34	15%
JEE	46	75%
Javascript	43	10%

According to these constants and the usage percentage of the language, the total lines of code to write are:

$$Size = Source\ Lines\ of\ Code = H_t \cdot (C_{\mathrm{HTML}} \cdot \mathrm{P}_{\mathrm{HTML}} + C_{\mathrm{JEE}} \cdot \mathrm{P}_{\mathrm{JEE}} + C_{\mathrm{JS}} \cdot \mathrm{P}_{\mathrm{JS}}) \tag{2.1}$$

Where:

- $\mathrm{C_i}$ express the *conversion factor* of the i language
- $\mathrm{P_i}$ express the *usage percentage* of the i of the language in the whole project
- $\mathrm{H_t}$ is the *FPs result*

The result is so found:

$$Size = Source\ Lines\ of\ Code = 184 \cdot (34_{\mathrm{HTML}} \cdot 0.15_{\mathrm{HTML}} + 46_{\mathrm{JEE}} \cdot 0.75_{\mathrm{JEE}} + 43_{\mathrm{JS}} \cdot 0.1_{\mathrm{JS}}) = \boldsymbol{8078} \tag{2.2}$$

2.2 Cost and effort estimation: COCOMO II

This section deals with the estimation of the cost and the effort that are required to develop the system. The evaluation will be carried out using the *COnstructive COst MOdel II* (later on referred as *COCOMO II*) and will make use of the estimated size of the project obtained in the previous section. Let us first introduce the empirical formula that will guide the effort estimation process:

$$Effort = A \cdot Size^{E} \cdot \prod_{i=1}^{17} EM_i \tag{2.3}$$

where:

- Effort is expressed in PM (Person-Months)
- $A = 2.94$ is a productivity constant in PM/KSLOC (Person-Months/Kilo-Source Lines of Code)
- Size is the estimated size of the project in KSLOC obtained in the previous analysis
- E is an aggregated index of five *Scale Factors*
- EM_i stands for *Effort Multiplier* and corresponds to the value of a specific entry that will be explained later in the *Cost Drivers* subsection

Next subsections are focused on the evaluation of the two last operands of the previous formula. At the end of the section the final result obtained applying formula 2.3 is shown.

2.2.1 Scale Factors

The first step consists in evaluating the actual value of exponent E in formula 2.3. To do so, a list of *Scale Factors* have to be computed. These entries represent an heterogeneous set of variables and their values are assigned according to the following table:

Scale Factor	Very low	Low	Nominal	High	Very high	Extra high
PREC	Thoroughly unprecedented	Largely unprecedented	Somewhat unprecedented	Generally familiar	Largely familiar	Thoroughly familiar
SF_i	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	Rigorous	Occasional relaxation	Some relaxation	General conformity	Some conformity	General goals
SF_i	5.07	4.05	3.04	2.03	1.01	0.00
RESL	Little (20%)	Some (40%)	Often (60%)	Generally (75%)	Mostly (90%)	Full (100%)
SF_i	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	Very difficult interactions	Some difficult interactions	Basically cooperative interactions	Largely cooperative	Highly cooperative	Seamless interactions
SF_i	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	Level 1 Lower	Level 1 Upper	Level 2	Level 3	Level 4	Level 5
SF_i	7.80	6.24	4.68	3.12	1.56	0.00

Let us discuss each of these entries:

- **Precedentedness**: it is the index that defines the previous experience of the team in developing such projects. This is the very first project, thus a *Very low* level is reasonable.
- **Development flexibility**: the index of the flexibility for the project choices with respect to the external interfaces and the requirements. The project has strict requirements to abide by in terms of functionalities of the system, but no specific architecture is required. This variable is then assigned a level of *Nominal*.
- **Architecture and risk resolution**: it reflects the capability of managing possible risks with an accurate contingency plan and the fact that a reasonable schedule is sketched. In the document both these aspects are covered in detail, so for this variable the right choice is to pick the *Very High* level.
- **Team cohesion**: it deals with the capability of the members of the team to work together in harmony and sharing the same vision. The team is made of only three people in this project and they work with great commitment and together, thus *Very high* is the right degree.

- **Process Maturity**: it is a index of the maturity of the organization. Even though this is the very first project of the team, the schedule has been accurately defined, the project widely documented and discussed among the members of the team. Consequently *Nominal* level is reasonable.

Here is the re-cap of the *Scale Factors* analysis:

Scale Factor	*Level*	*Value*
Precedentness (PREC)	Very low	6.20
Development flexibility (FLEX)	Nominal	3.04
Architecture and risk resolution (RESL)	Very high	1.41
Team cohesion (TEAM)	Very high	1.10
Process Maturity (PMAT)	Nominal	4.68
Total sum		16.43

It is now possible to evaluate exponent E with the following formula:

$$E = 0.91 + 0.01 \cdot \sum_{i=1}^{5} SF_i$$

The result is

$$E = 1.0743$$

2.2.2 Cost Drivers

After the evaluation of exponent E, it is time to compute the *Effort Multipliers* factors EM_i in formula 2.3. The values of these parameters are assigned with the help of these standardized tables:

- **Required Software Reliability**

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

Figure 2.1: RELY Cost Driver

The system is in charge of managing data inside the car that can potentially lead to major issues (inability to open the car, incorrect battery level visualization) but not so dangerous for human life. Thus, only financial losses related to clients dissatisfaction are to be considered in this context and a *Nominal* level is reasonable.

- **Database Size**

DATA* Descriptors		Testing DB bytes/Pgm SLOC < 10	10 ≤ D/P < 100	100 ≤ D/P < 1000	D/P ≥ 1000	
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

Figure 2.2: DATA Cost Driver

This is an index of the effort needed to maintain the data required to complete the test of the project. Since the estimated project size is not so large (P = 8078 lines of code) and for testing purpose only few MB of storage are necessary (let's say D = 2 MB), the $D/P = 248$ ratio indicates that for this cost driver the level has to be set to *High*.

- **Product Complexity**

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

Figure 2.3: CPLX Cost Driver

This index is the average of the values of five different entries: control operations, computational operations, device-dependent operations, data management operations and user interface management operations.
Control operation estimates the complexity of the operators and the methods used in each component and the calls from one module to another one. At this level only the higher-level architecture is designed, so the real implementation is not available, but considering the expertise of the work team a *Nominal* level is reasonable.
Computational Operations estimates the complexity of the mathematical expression used in the project and for the same reason explained in the previous entry a *Nominal* level is chosen.
Device-dependent Operations is related to the management of the I/O operations. There are different types of operations of this kind (i.e the management of the recharge of a car) and the I/O processing of course includes device selection, status checking and error management. This corresponds to a *Nominal* level too.
Data Management Operations estimates the complexity in the data management from the data structure handled by the system to the management of data within the Database. Quite complex structures are needed to manage information such as car positions and map services. For this reason the level of this entry should be set to *High*.
User Interface Management Operations estimates the complexity of the User Interface of the system. A simple but fancy and intuitive interface will guide users of the system in performing the desired operations. *Low* level is here appropriate.
As mentioned above CPLX Cost Driver is set to the average of these values: *Nominal* is the result.

- **Developed for Reusability**

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

Figure 2.4: RUSE Cost Driver

This Cost Driver accounts for the reusability of the components of the system. Since this project and its modules are inteded as a standalone work a *Low* level is chosen.

- **Documentation Match to Life-Cycle Needs**

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 Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.81	0.91	1.00	1.11	1.23	n/a

Figure 2.5: DOCU Cost Driver

This parameter describes how well the documentation deals the application requirements. The documentation provided catches every need of the product life-cycle, thus a *Nominal* level is the right choice.

- **Execution Time Constraint**

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 Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	n/a	1.00	1.11	1.29	1.63

Figure 2.6: TIME Cost Driver

This is a measure of the expected amount of CPU usage with respect to the hardware capability. Since the system is constantly in connection with the cars and chances are that many users are requesting services at the same time a *Very high* level is chosen.

- **Main Storage Constraint**

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

Figure 2.7: STOR Cost Driver

This is an index of the cost required to store the software program on the main storage. This is a very little concern since nowadays the cost related to storage is very low. *Nominal* level is chosen.

- **Platform Volatility**

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

Figure 2.8: PVOL Cost Driver

This parameter stands for the frequency of changes and updates in the platform (i.e software and hardware) of the system. Since the core of the system is developed with a well-known, stable and standardized infrastucture (JEE mainly) few updates are likely to be applied. For this reason a *Low* level is reasonable.

- **Analyst Capability**

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	n/a

Figure 2.9: ACAP Cost Driver

This is an index of the ability, efficiency and commitment of the analysts (that in this case coincide with the team work) to properly understand the problem with respect of the real world in which the system will work. The members of this team adopted a systematic way in dealing the different aspects of the project with a complete documentation as a result. The level of this parameter is consequently set to *High*.

- **Programmer Capability**

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	n/a

Figure 2.10: PCAP Cost Driver

This parameter catches the capability of the programmers in the sense of being able to work as a team rather than as individuals. Communication, Cooperation and thoroughness have been constant throughout the development. A *High* level is suitable here.

- **Personnel Continuity**

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	

Figure 2.11: PCON Cost Driver

This index informs about the project's annual personnel turnover: 3% is a very high continuity while 48% a very low continuity. For what concerns this project the three members of the team worked thoroughly for the entire project. Thus, *Very high* level is the right choice.

- **Application Experience**

APEX Descriptors:	≤ 2 months	6 months	1 year	3 years	6 years	
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

Figure 2.12: APEX Cost Driver

Even though the team has a basic knowledge of software program development, its experience in implementing application with a complexity similar to the one of the PowerEnJoy is quite low. For this reason the chosen value for APEX driver is *Very Low*.

- **Platform Experience**

PLEX Descriptors:	≤ 2 months	6 months	1 year	3 years	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

Figure 2.13: PLEX Cost Driver

Since this is one of the firsts experience for the team in developing such a big and quite complex system, its background in using specific platform to properly manage and sustain software of this dimension is low. Thus a *Very Low* level seems a reasonable choice.

- **Language and Tool Experience**

LTEX Descriptors:	≤ 2 months	6 months	1 year	3 years	6 year	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.20	1.09	1.00	0.91	0.84	

Figure 2.14: LTEX Cost Driver

The team has an intermediate level background of Java language but it is not very skilled in JavaEE development; moreover it has already used different tools for both code and documentation management. The experience has been acquired mostly in a project that left about 6 months, then the value *Low* is the one that better catch this situation.

- **Use of Software Tools**

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 Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.17	1.09	1.00	0.90	0.78	n/a

Figure 2.15: TOOL Cost Driver

The specific tools that will be used during the implementation phase allows a basic life-cycle management and it is moderately integrated; according to the previous table the chosen level is then *Nominal*.

- **Multisite Development**

SITE: Collocation Descriptors:	Inter-national	Multi-city and Multi-company	Multi-city or Multi-company	Same city or metro. area	Same building or complex	Fully collocated
SITE: Communications Descriptors:	Some phone, mail	Individual phone, FAX	Narrow band email	Wideband electronic communicat ion.	Wideband elect. comm., occasional video conf.	Interactive multimedia
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.22	1.09	1.00	0.93	0.86	0.80

Figure 2.16: SITE Cost Driver

Even though the team members live in different cities, the internal communication has been carried using highly interactive multimedia applications that have hide the distance factor and made the information and files exchange very fast and efficient. The appropriate level for this driver is then *Extra High*.

- **Required Development Schedule**

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

Figure 2.17: SCED Cost Driver

The effort invested into the preparation of the various document has been sufficient to redact all the parts and revise them accordingly; furthermore the effort has been distributed quite equally in every time interval to produce each document. For these reason the chosen level is *Nominal*.

A summary of all the taken decisions for each cost driver is reported in the following table:

Cost Driver	*Level*	*Value*
Required Software Reliability (RELY)	Nominal	1.00
Database size (DATA)	High	1.14
Product complexity (CPLX)	Nominal	1.00
Required Reusability (RUSE)	Low	0.95
Documentation match to life-cycle needs (DOCU)	Nominal	1.00
Execution Time Constraint (TIME)	Very High	1.29
Main storage constraint (STOR)	Nominal	1.00
Platform volatility (PVOL)	Low	0.87
Analyst capability (ACAP)	High	0.85
Programmer capability (PCAP)	High	0.88
Personnel continuity (PCON)	Very High	0.81
Application Experience (APEX)	Very Low	1.22
Platform Experience (PLEX)	Very Low	1.19
Language and Tool Experience (LTEX)	Low	1.09
Usage of Software Tools (TOOL)	Nominal	1.00
Multisite development (SITE)	Extra High	0.80
Required development schedule (SCED)	Nominal	1.00
Total product		0.9323

2.2.3 Effort estimation

After the *Scale Factors* and *Cost Drivers* analysis which lead to the evaluation of the exponent E and of the EM_i (*Effort Multipliers*) factors of formula 2.3 it is now possible to estimate the $Effort$ required by the project:

$$Effort = 2.94 \cdot (8.078)^{1.0743} \cdot 0.9323 \approx 26 \quad PM$$

2.2.4 Schedule estimation

Another interesting estimation is about the *Duration* that will be needed in order to complete the project. Let us introduce the following formula:

$$Duration = 3.67 \cdot Effort^F \tag{2.4}$$

where exponent F is computed as follows:

$$F = 0.28 + 0.2 \cdot (E - B) = 0.28 + 0.2 \cdot (1.0743 - 0.91) = 0.31286$$

Applying the actual values of the parameters formula 2.4 yields

$$Duration = 3.67 \cdot 26^{0.31286} = 10.17 \quad months$$

3 | Schedule

A fundamental part of the *Project Plan* is the scheduling. This step allows to divide the overall work, that has to be done, in separated tasks and moreover to define when and how they will be completed; in particular the estimation of the calendar time required to execute each task is provided through the Gantt diagram.
It should be noted that the time estimation of the tasks is not trivial and while doing this it's important to consider that the productivity is not proportional to the number of people of the team and that some extra time must be added because sometimes the flow of the events does not follow the expected one.
The analysis and the definition of the various tasks is carried out at a quite coarse grained level thus only the main activities are included into the Gantt diagram, however they shouldn't be too small or too big; for instance the length ranges between a few days and 2 weeks.
Lastly, the start of the schedule of the project is set to the *October* 16^{th}.

The following *Figure* reports the overall schedule of the main phases of the project, this lets to have a simple, fast and readable feedback of the time effort required from each phase.

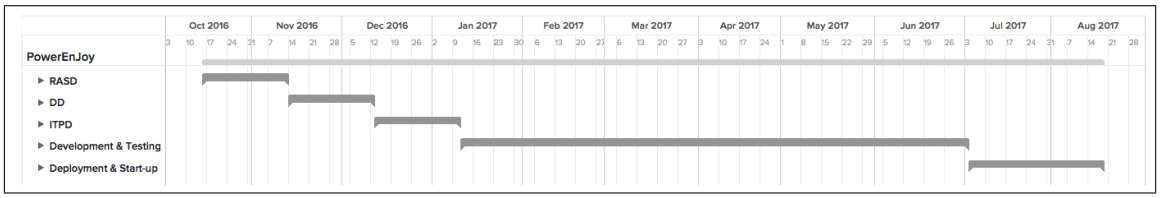

Figure 3.1: Overall schedule

The details and tasks partitioning for every phase are reported in the following *Figures*.

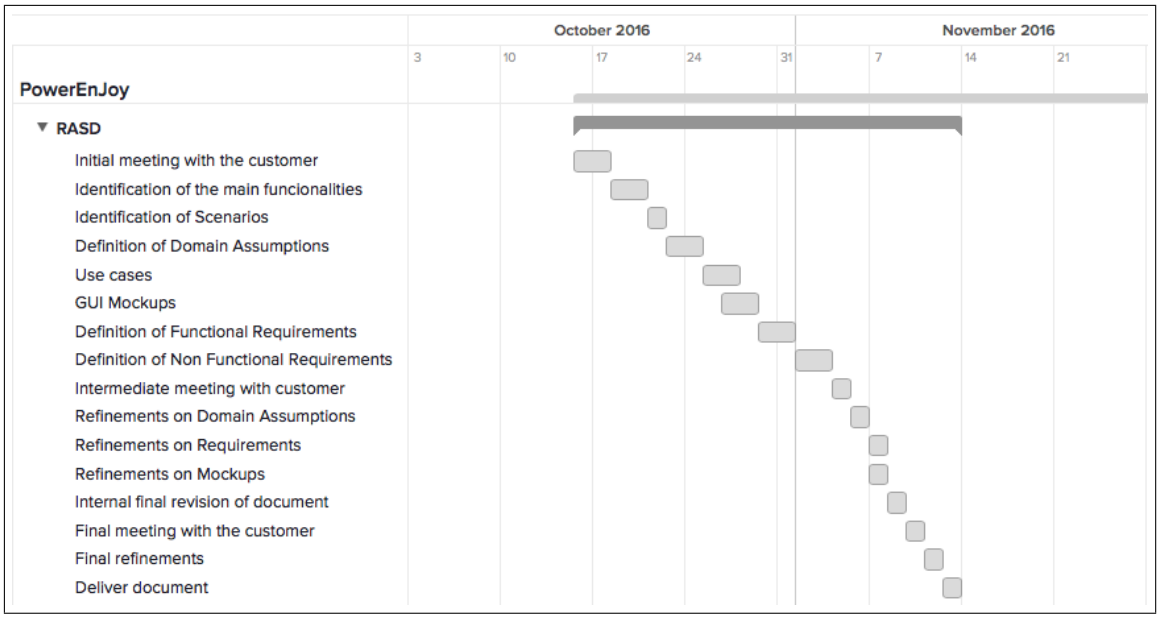

Figure 3.2: RASD schedule

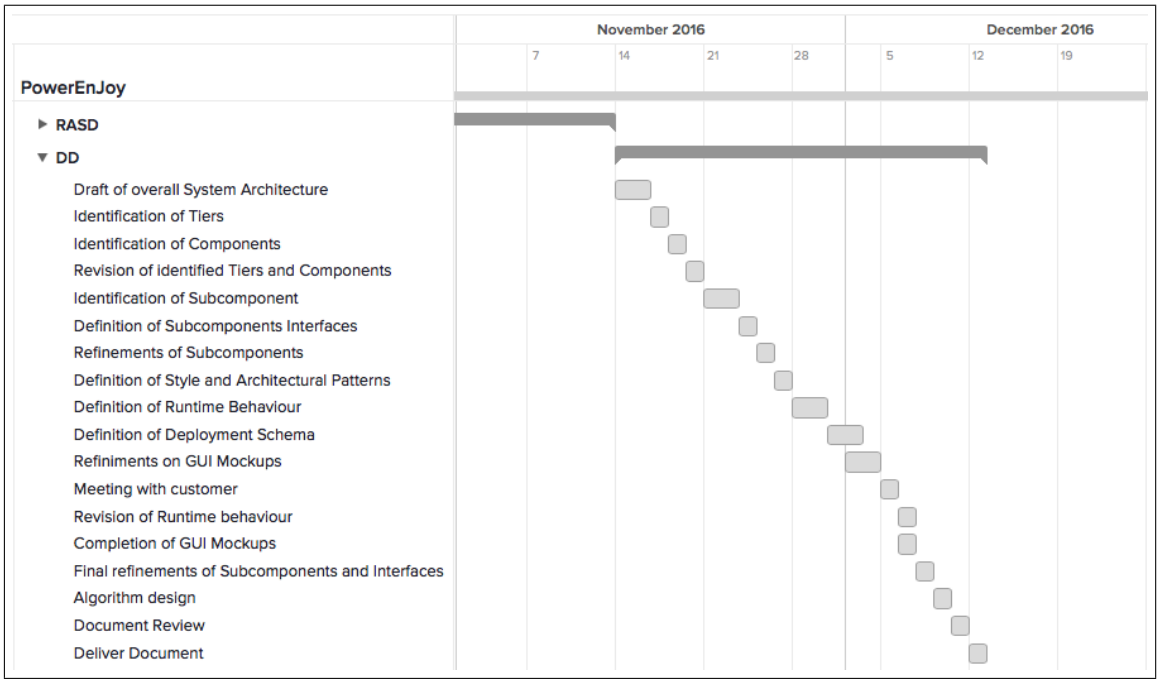

Figure 3.3: DD schedule

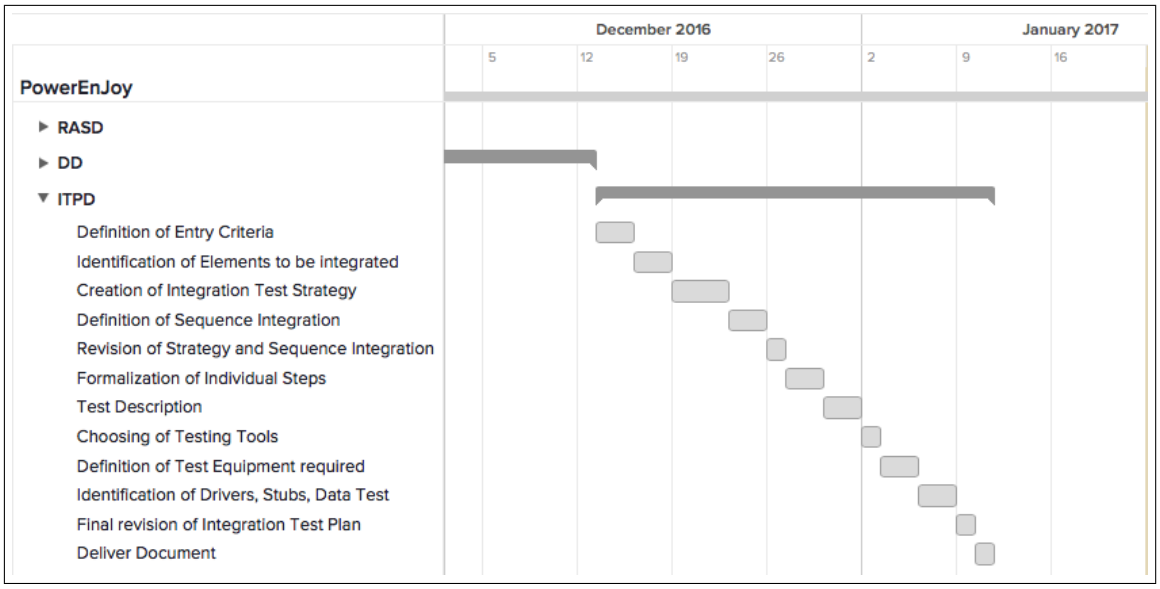

Figure 3.4: ITPD schedule

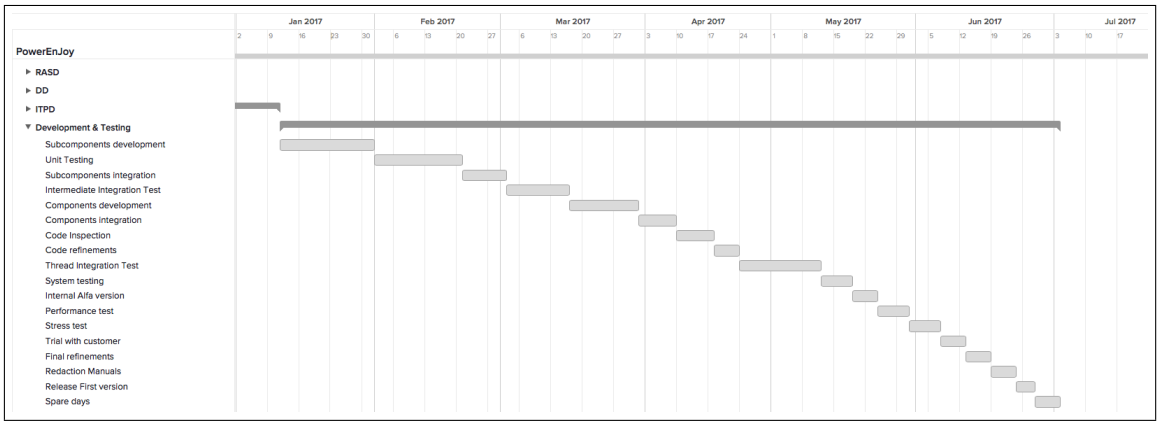

Figure 3.5: Development & Testing schedule

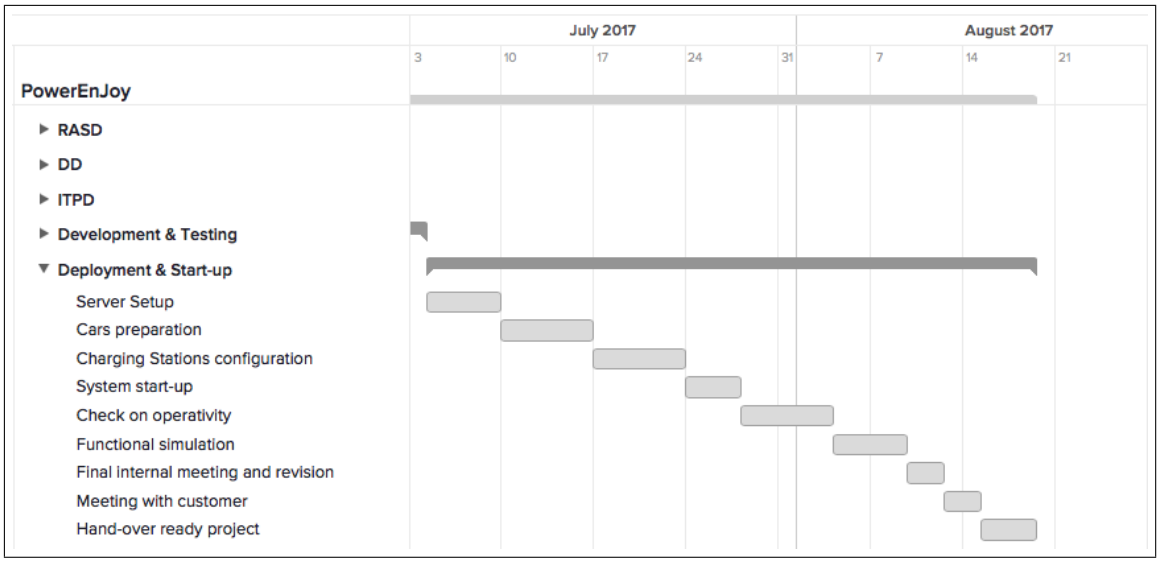

Figure 3.6: Deployment & Start-up schedule

4 | Resource Allocation

The Resource Allocation phase is strongly correlated with the Schedule phase because its main goal is to properly assign every single task, defined in the previous step, to the components of the development team. The assignment process should be done carefully and in such a way to optimize the effort of each team member; for instance, the allocation of all the activities and tasks is done with respect to the three authors of these documents assuming that will be also the development team.
The chosen allocation is provided via a simple double-entrance table that shows the assignment for each task with respect to team members. For readability reasons, the table has been split in two parts.

	Development Team		
	Melloni Giulio	Renzi Marco	Testa Filippo
RASD			
Initial meeting with the customer	•	•	•
Identification of the main functionalities	•	•	•
Identification of Scenarios	•	•	•
Definition of Domain Assumption	•	•	•
Use cases		•	•
GUI mockups	•		
Definition of Functional Requirements		•	•
Definition of Non Functional Requirements	•		
Intermediate meeting with customer	•	•	•
Refinements on Domain Assumption	•	•	•
Refinements on Requirements		•	•
Refinements on Mockups	•		
Internal final revision of document	•	•	•
Final meeting with the customer	•	•	•
Final refinements	•	•	•
Deliver document	•	•	•
DD			
Draft overall system architecture	•	•	•
Identification of Tiers	•	•	•
Identification of Components	•	•	•
Revision of identified Tiers and Components	•	•	•
Identification of subcomponents	•	•	•
Definition of subcomponents interfaces	•	•	•
Refinements of subcomponents	•	•	•
Definition of styles and architectural patterns	•	•	•
Definition of runtime behaviour	•	•	
Definition of deployment scheme			•
Refinements of GUI mockups	•		
Meeting with customer	•	•	•
Revision of runtime behaviour		•	•
Completion of mockups	•		
Final refinements of subcomponents and interfaces	•	•	•
Algorithm design	•	•	•
Document review	•	•	•
Deliver document	•	•	•
ITPD			
Definition of entry criteria	•	•	•
Identification of elements to be integrated	•	•	•
Creation of integration test strategy	•	•	•
Definition of sequence integration	•	•	•
Revision of strategy and sequence	•	•	•
Formalization of individual steps	•	•	•
Test description	•	•	•
Choosing of testing tools		•	
Definition of test equipment	•		•
Identification of drivers, stubs and data test	•	•	•
Final revision of integration test plan	•	•	•
Deliver Document	•	•	•

Figure 4.1: Resource allocation - part 1

	Development Team		
	Melloni Giulio	Renzi Marco	Testa Filippo
Development & Testing			
Subcomponents development	•	•	•
Unit testing	•	•	•
Subcomponent integration	•	•	•
Intermediate integration test	•	•	•
Components development	•	•	•
Components integration	•	•	•
Code inspection	•	•	•
Code refinements	•	•	•
Threat integration test	•	•	•
System testing	•	•	•
Internal alfa version	•	•	•
Performance test	•	•	•
Stress test	•	•	•
Trial with customer	•	•	•
Final refinements	•	•	•
Redaction manuals	•	•	•
Release first version	•	•	•
(Spare days)			
Deployment & Start-up			
Server setup	•	•	•
Cars preparation	•	•	•
Charging stations configuration	•	•	•
System start-up	•	•	•
Check on operativity	•	•	•
Functional simulation	•	•	•
Final internal meeting and revision	•	•	•
Meeting with customer	•	•	•
Hand-over ready project			

Figure 4.2: Resource allocation - part 2

5 | Risk Management and Mitigation

This section deals with the possible risks that can threaten the system during its development process. These hazards can stem from a variety of sources and thus can be classified as:

- **Project risks**: if they threaten the project plan, such as the project schedule. As can be seen in the following tables, the most common problem that can arise from such risks is a delay in the project release.
- **Technical risks**: they are related to the technical part of the project. They include a major variety of problems such as unskilled staff, flaws in the external adopted components, changes in the set of the requirements and so on.
- **Business risks**: they include a set of heterogeneous hazards such as budget cuts, sales falls and market policy flaws. If they become concrete, they can compromise the viability of the project.
- **Personnel risks**: this type of hazards deals with all the possible problems that can be met within the team of work.

Since a pro-active risk-management approach is desirable, for each category of risk a list of the most recurrent hazards will be taken into account, along with their probabilities to be faced, their levels of impact on the project and the strategies that can be adopted to mitigate them.

5.1 Project risks

Risk	Probability	Impact	Strategy
Underestimated development time	Moderate	Serious	If the spare time is not sufficient, try to negotiate with the customer the possibility to have two releases, the first to the planned one and the second one as early as possible.
A change in the direction of the project	Low	Moderate	Redact very precise and detailed documentation of the project so that new managers are able to handle the process.

5.2 Technical risks

Risk	Probability	Impact	Strategy
Difficulty in recruiting a skilled staff	Moderate	Serious	Be prepared and inclined to pay extra to find skilled people. If it is not sufficient, may consider the possibility of buying external components that are already developed.
Changes in the requirements	Low	Serious	Be compliant with the requirements traced in the *RASD* and in case changes occur evaluate the trade off related to the needed variations in the project. Increment on the price should come from the customer.
External components have flaws	Moderate	Serious	Choose components that are on the market for a long time, so that they can be reliable. Experience with precedent projects may be an advantage in this case.
Modification in the APIs of the external components	Low	Moderate	Write the code as portable as possible. Changes in the APIs should not break the system. Also plan updates in case such modifications occur.

5.3 Business risks

Risk	Probability	Impact	Strategy
Budget cuts during the development	Moderate	Tremendous	This kind of risk is the most critical one. In order to mitigate it, redact a document in which the financial benefits associated to the business of the project is clearly stated and try to convince managers not to apply cuts.
Poor sales	Moderate	Serious	Make special discounts and offers at the launch of the product in order to achieve great popularity. In doing so, a market analysis has to be carried out .
Competitors in the market	Moderate	Moderate	Choose a guide line for your product: should it be cheaper than the other or should it aim to optimality? In the launch phase also consider the possibility of making special discounts to attract clients.

5.4 Personnel risks

Risk	Probability	Impact	Strategy
Poor motivation	Moderate	Moderate	Try to select people that are skilled in particular fields and have experience in their work. Praise their commitment with rewards.
Conflicts within the work team	Low	Moderate	Promote the cooperation of people by making a community that periodically meets and discuss on the level of the project.
Key staff are ill at critical times in the project	Moderate	Serious	Consider the possibility to have several people working on common tasks in order to prevent excessive time losses.

6 | Effort Spent

In order to complete this document, each author worked for XX hours.