



POLITECNICO
MILANO 1863

Project Plan Document

Guido Muscioni (mat. 876151)

Marco Orbelli (mat. 876649)

Paola Marchesini (mat. 876541)



Figure 1: New brand logo.

Contents

1	Introduction	3
1.1	Purpose	3
1.1.1	Scope	3
1.2	List of Definitions and Abbreviations	3
1.2.1	Definitions and abbreviations	3
1.3	List of Reference Document	4
2	Project Size, Cost and Effort Estimation	5
2.1	Size estimation: Function Points	5
2.1.1	Internal Logic Files	5
2.1.2	External Logic Files	6
2.1.3	External Inputs	7
2.1.4	External Inquiries	9
2.1.5	External Outputs	9
2.1.6	Overall Estimation	9
2.2	Cost and Effort Estimation: COCOMO II	9
2.2.1	Scale Factors	9
2.2.2	Cost Drivers	9
2.2.2.1	Product Factors	9
2.2.2.2	Platform Factors	11
2.2.2.3	Personnel Factors	12
2.2.2.4	Project Factors	14
2.2.2.5	General Factors	15
2.2.2.6	Final result	15
3	Schedule	17
4	Resource Allocation	18
5	Risk Management	19
6	Appendices	20
6.1	Used Tools	20
7	Hours of Work	21

1 Introduction

1.1 Purpose

<TODO: Guido>

1.1.1 Scope

The scope of that system, is defined in the RASD document, then, here there is a copy of the paragraph:

The aim of the project is to develop a digital system for an electric car-sharing company, that is called PowerEnjoy. There are not previous system so that document will describe the requirements that deal with all the system that the customer wants.

The users of PowerEnjoy system are:

- *Client;*
- *Employee.*

Both of these users are going to have unique credentials in order to use the system, so they must be registered.

The clients will use this system to reserve and use cars of PowerEnjoy company.

The employees will use that system in order to know what cars need help.

1.2 List of Definitions and Abbreviations

1.2.1 Definitions and abbreviations

- **SDD**: Software Design Description.
- **DD**: Design Document.
- **ITP**: Integration Test Plan.
- **ITPD**: Integration Test Plan Document.
- **DBMS**: Database Management System.
- **ER**: Entity-Relationship diagram.
- **RASD**: Requirement Analysis and Specification Document.
- **SDK**: Software Development Kit.
- **PhoneGap**: mobile cross-platform development framework.

- **Dispatcher:** The dispatcher is a component devoted to forward requests received from the user terminal to the right controller. It is capable to understand what it has to do, according to informations it received. It does not do anything excepts forwarding, for this reason it can be considered a mediator between the smarter part of the application server and humans.
- **Framework:** it is a set of functions that support the developers during developing and integration phase.
- **OS:** Operating System.
- **Id:** An identifier is a name that identifies a unique element.
- **RAM:** Random Access Memory is a form of computer data storage which stores frequently used program instructions to increase the general speed of a system.
- **MB:** Megabyte (10e6)
- **Type of test:**
 - D: test with Data Access Interface
 - I: integration test between controllers
- **JPA:** Java Persistence API.
- **JavaEE:** Java Enterprise Edition.
- **RESTful :** REpresentational State Transfer is an architectural style and an ap- proach to communications (often used in the development of Web services);
- **API:** Application Programming Interface.
- **Req** as for Requirement.
- **WebApp** as for Web Application.
- COCOMO II
- COCOMO II document

1.3 List of Reference Document

- Specification document: Assignments AA 2016-2017;
- Examples documents:
 - Sample Integration Test Plan Document.

2 Project Size, Cost and Effort Estimation

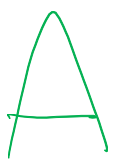


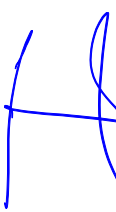
<TODO: function point approach and COCOMO>

2.1 Size estimation: Function Points

This section is built using the Function Points approach to evaluate the complexity and the quality of the developed software. This approach provides a dimensionless parameter to evaluate the functionalities of the software. The Function Points reduces the abstraction of the product, giving concrete datas to evaluate the project. The tables below represents the Function Points value that are used in order to assign complexity held to our files and other parts of the project.

2.1.1 Internal Logic Files

The PowerEnjoy system needs to store in its DataBase some informations about its users and in general about the service it provides in order to ensure a good service. In this section, there is a detailed analysis about the Internal logic files (ILFs) needed to the functionalities of the system.

- User data: first the system has to know the information about its users (clients who use the PowerEnjoy application). The DB has a table in which are stored the data required at the registration phase: name, surname, password, driving license, email and the telephone number. The system must acquire the payment information at the “sign up” , in order to guarantee the functionality of automatic payment of the ride. This last information is stored in a different table from the one explained before, in which are stored only general data about the users. 
- Employee data: the employee data are stored in two different tables. The first table contains the general information of an employee, like the SSN number, name, surname, email address and the password of his/her account. The second table. Instead, the second tables contains information about the phone provided by the company, in particular the IMEI code (International Mobile Equipment Identity) to identify the physical device and the telephone number associated to it. It has been chosen to store the phone information in a different table, because the company may not provide a phone for every employee, so it is necessary to underline that these devices only could be used by an employee and, at the same time the company must find all the free devices. 
- Hystorical work data: the data about the “story” of employees’ works are stored in a single table with a key ID, where it can be found detailed information of a work: date, time, car involved, couple of employees that have done the work, a brief description and a flag ~~due to~~ to understand if the phase of the car involved is changed. The reason of that choice (use of a flag) is that an employee could remove a car from his/her list due to force majeure, left the work undone. 
- List data: the list data are an important part of our system. In this table are stored the necessary information in order to identify a list of a car: id of the list, and the two employees whose take care of the current list. The cars that are included in the list are stored in another table so the sysem has to execute a query on the id of the list, which a car also has, to found all the members of the list. 

- Car data: this internal files contain ~~all~~ the information about the car which are managed by the PowerEnjoy system: code (key of the table) used by users to open/close the car, license plate, battery level, number of seats ~~used~~ and the information to identify the position of the car (longitude and latitude). The information about the phase of the car (so the state in which it is) are stored in a different table, containing the id of the car and its actual phase. ~~The use of two tables is justified by a simpler management of the queries which are interested only in the phase of the car.~~
- Reservation data: the information about a reservation are stored in a table containing the id of the reservation, the date, the initial time and the final time. To obtain details about the user who have done the reservation or the ride connected to it, it is necessary to execute queries that involved other data structures.
- Ride data: the ride data structure uses a two-level structure. In the first table there are the general information to identify a ride: id, starting and final longitude and latitude and the number of passengers. The second table of this data structure contains the information about the payment of the ride (id, time, price, type of discount, quantity of discount).
- Area data: this data structure contains two tables in order to distinguish the different area that are managed by the PowerEnjoy system. The first table contains information about the special areas, so latitude and longitude to identify the center of the area, the radius and the plugs which are available in this zone. The second table contains information about the safe area external point: <TODO paola + guido : dire i campi (latitude, longitude, order) e spiegare a cosa si riferiscono e cosa significano>.

(Note: this section is extremely connected with the entity-relationship diagram, that is contained in the DD document.)

2.1.2 External Logic Files

In this section all the external sources that are used by the system will be analyzed with respect to the function points approach. According to the previous documents the external sources are:

- Google Map service;
- SMS service;
- Mail service;
- Payment service (through PayPal).

However the Mail service does not need any kind of files, in actual fact the system sends directly the mail with an automated text.

Google Map service The system will be integrated with Google Map through Google Map API. The data returned from Google are GeoJSON vectors, a format used to save and exchange geospatial data over the internet. That service is used for:

- Build the path that connect the position of a car to the position of an user;

- Create the map in which the car are visualized at the moment of the reservation.

Our system will use GeoJSON Google implementation in order to treat that data, and the complexity level is set to an average level.

SMS service The system is also integrated with an SMS service in order to perform the notifications and the commands defined in the first part of the project. SMS general API will be used in order to perform the integration with the SMS provider. The system will treat with XML SMS type files in order to:

- Receive an SMS and use that to do action on the car.

We have never used that kind of APIs, but, reading the documentation, we decide to give a medium value of complexity to that part of the system.

Payment service Following the PayPal API documentation the system will be integrated with that service with a built-in document file defined by PayPal: PDT (Payment Data Transfer). The action of the system that treat that type of documents a quite complex, beacuse the algorithms must be very secure and optimized. The PDT document will be used in order to:

- Save the payment of a ride;
- Take the aknowledge of the symbolic withdraw at the registration phase.

According to what is mentioned in that paragraph the level of complexity is set to high.

<TODO: mettere quei cazzo di function points.>

2.1.3 External Inputs

<TODO: tutte le operazioni che possono fare gli user, sono i metodi dei diagrammi UX>

The PowerEnjoy system provides a great number of functionalities which support interactions with people who use the application or the web site. In this section the various interactions are explained and grouped by type of user:

General user:

- Sign in: this operation is simple because it involved only the account manager, which has to verify the credential used to sign in. It yelds... 3
- Sign up: this operation has an average complexity because the system has to verify all the credentials inserted by the user and especially the payments information (with a symbolic withdraw). It yelds... 4

User verified:

- Look for car nearby: this operations has an hight complexity because it is a query on the DB, searching for the car nearby the position detected or inserted by te user. It also upload all the information about the cars returned. It yelds... 6
- Insert postition: this is a simple operation, the user insert manually his position. It yelds... 5

- Select the car: given that the upload of the car informations it was already done, this operation simply visualize the information of the car selected. It yields...
- Reserve that car: this operation has an high complexity because it involves a lot of component which have to communicate and to query the DB uploading it. It also involves the notification to the users. yields...
- Cancel the reservation: this is an operation of average complexity because it calls a controller which update the DB, and then, with a notify to the car controller, the table of cars are also updated. It yields...
- See the way to the car: this operation has an average complexity because it requires the interaction with the map gateway to compute the best way to reach the car and also the elaboration of data send to the system from the gateway. It yields...
- Open the car:
- Modify: this operation has an average complexity (according to the "Sign up" operation) because the system has to verify the new credentials inserted by the user. It yields...

Employee:

- Look for my list: this operation has an high complexity: it necessary to do a complex query on the db, lookinh for the car that needs attention. The list contains also all the informazione It yields...
- See the state of the car: given that the upload of the car informations it was already done in "Look for my list", this operation simply visualize the information of the car selected. It yields...
- See car in a map:
- Change state of car: this operation has an high complexity. When an employee ends its work, it has to change the phase of the car. This functionality includes both queries on the DB and interactions between the controllers, in order to change the phase and also to remove the car, updating the tables involved. It yields...

Car driver:

- Skip that page: this operation is simple, it only redirects the user to another page, without particular query or interaction with the components of the system. It yields...
- Activate money saving option: this operation has an high complexity because it requieres queries on the DB to show the Safe Area nearby the destination of the user and also to compute the plug free. It also update the discount that must to be applied at the end of the ride. It yields...
- Call / Terminate the call: this is a straighforward operation, it simply call the service center thanks to the mobile device included in the car system. It yields...
- Finish the ride: this operation is a strightforwar operation, in actual fact this functionality update only a table of the DB setting the state of the ride. It yields...

2.1.4 External Inquiries

<TODO: sono le richieste degli utenti rivolte all'ottenimento di dati.>

2.1.5 External Outputs

The PowerEnjoy system needs also to interact directly with the user, send him/her mail to upload the state of his/her activities:

- Notification to the user to send his/her credentials.
- Mail to the user to send the bill of his/her ride.
- Notification send to the user with the reservation message.

2.1.6 Overall Estimation

<TODO: all together.>

The following table show the results of our estimation:

2.2 Cost and Effort Estimation: COCOMO II

2.2.1 Scale Factors

2.2.2 Cost Drivers

2.2.2.1 Product Factors

Required Software Reliability (RELY) For PowerEnjoy company, one or more disruptions and a possible breakdown of the system, can lead, firstly in an high financial loss. Secondly, in the meantime of the problem, the users who are using the system will not clearly be satisfied. According to the rule of COCOMO II, that driver is set to high.

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

Table 1: RELY cost driver.

DataBase Size (DATA) <TODO: dopo che marco mi dice quante linee di codice massime e quante linee di codice minime devo applicare la formula d/p>

DATA Descriptors:		Testing DB bytes/Pgm SLOC < 10	$10 \leq \frac{D}{P} < 100$	$100 \leq \frac{D}{P} < 1000$	$\frac{D}{P} \geq 1000$	
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

Table 2: DATA cost driver.

Product Complexity (CPLX) Using the model definition manual of COCOMO II, we analyzed all the five areas that take part in complexity estimation:

- **Control Operations:** the code that will be produced is based on MVC pattern with Observable/Observer interfaces. So that it will not be simple, furthermore the system must control real-time the lists and the cars itself. According to COCOMO II the level is set to high;
- **Computational Operations:** following the choices that we have taken in the previous document, the level of that area is set to high. In actual fact, in order to calculate if a car is in a safe area the system must be a kind of interpolation between the external points saved in the DB;
- **Device-dependent Operations:** the nominal value fit quite well in that case, infact the system has been built on the state checking of the cars, but the timing of that operations can be quite variable;
- **Data Management Operations:** the database is extremely use, and important, in the system, however it does not need a complex search engine. Thanks to that facts, the level is set to high;
- **User Interface Management:** the user interface of the system is not very complex, it is designed in order to keep all the action flows very simple, so that the level is set to nominal.

Following what is mentioned before the general level of complexity is set to high.

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

Table 3: CPLX cost driver.

Developed For Reusability (RUSE) In the requirements analysis phase and in particular in the design phase we have focused on developing as much code as possible that can be shared between PowerEnjoy system (for example the list controller, and the phonegap application). Thus, the level of reusability is set to nominal.

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

Table 4: RUSE cost driver.

Documentation Match to Life-CycleNeeds (DOCU) Thanks to the document produced, and the algorithm description of DD document, we think that a normal level of documentation will be sufficient in order to maintain the code in the future, so the level of DOCU is nominal.

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

Table 5: DOCU cost driver.

2.2.2.2 Platform Factors

Execution Time Constraint (TIME) That driver shows the weight on the processors of PowerEnjoy system. As we decided to build a server farm, even if the system is quite complex, as we have mentioned before, the level in this case is set to high.

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

Table 6: TIME cost driver.

Main Storage Constraint (STOR) That value represent the percentage of space that the system is going to use for its correct behaviour. As we would not waste a lot of resources the level of STOR is set to high.

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

Table 7: STOR cost driver.

Platform Volatility (PVOL) Following the principles that we have defined in previous document, the system must be up to date, but all the upgrades must be as robust as possible, in order to not introduce some critical bugs that can lead in a malfunctioning. So that the level is set to low.

PVOL Descriptors:		Major change every 12 months. Minor change every 1 month.	Major: 6 months; Minor: 2 weeks	Major: 2 months; Minor: 1 week	Major: 2 weeks; 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

Table 8: PVOL cost driver.

2.2.2.3 Personnel Factors

Analyst Capability (ACAP) As it can be seen from the previous document, we have conducted a complete and clear analysis of the scenarios in which PowerEnjoy system and the users can be involved. Moreover we have tried to considered as much critical cases as possible, in order to design some solutions, as the two server farms and the two different way of opening a car. Thus leads us to set the level of ACAP to high.

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

Table 9: ACAP cost driver.

Programmer Capability (PCAP) Following the rules defined by COCOMO II document, the level of that driver must be related to the “*capabilities of the programmer as a team. [...] ability,*

efficiency and thoroughness, and the ability to communicate and cooperate.” With respect to that description the PCAP is set to high, because as a team we are able to work and cooperate in a very good way. Obviously that is only a prediction, in actual fact we are not going to implement that project.

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

Table 10: PCAP cost driver.

Personnel Continuity (PCON) We do not have unlimited time to use for the developing of that project so we are going to set the level to low, according to the “*Hours of work*” reported in all previous documents.

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	

Table 11: PCON cost driver.

Applications Experience (APEX) We do not have a lot of experience in J2EE and also with the managing of the API that we have mentioned in the DD document, so that, the level of APEX is set to low.

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

Table 12: APEX cost driver.

Platform Experience (PLEX) We know something of all the platform that we have considered in order to simplify the building of that project. However our knowledge are set to a scholl level because we have used that platforms only during some university projects. According to that we decided to set PLEX to nominal.

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

Table 13: PLEX cost driver.

Language and Tool Experience (LTEX) What we have mention in PLEX and APEX paragraph is valid also for that driver. Not all the members of the team know how to write a PhoneGap application and the same for a real world database; so the level is set to low.

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	

Table 14: LTEX cost driver.

2.2.2.4 Project Factors

Use of Software Tools (TOOL) We have done the design of the project using some pretty software tools that can simplify and improve the developing of the project. Some examples are PhoneGap, in order to build a cross-platform application, and jersey framework, in order to easily integrate our J2EE architecture with the RESTful API. According to that facts the level of that driver is set to very high.

TOOL Descriptors:	edit, code, debug	simple, fronted, 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

Table 15: TOOL cost driver.

Multisite Development (SITE) Even though we live in different cities, we are placed in Milan in order to follow our courses in the university. Moreover we have always worked together in order

to have always a complete view of the project progress. So we decided to set that level to extra high.

SITE Collocation Descriptors:	Inter-national	Multi-city and Multi-company	Multi-city or Multi-company	Same city or metropolitan area	Same building or complex	Fully located
SITE Communications Descriptors:	Some phone, mail	Individual phone, FAX	Narrow band email	Wideband electronic communication	Wideband elect. comm., occasional video conf	Interactive multimed
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

Table 16: SITE cost driver.

2.2.2.5 General Factors

Required Development Schedule (SCED) According to “*Hours of work*” reported in all previous documents and the general effort that we are spending in the building of all documents, the SCED level is set to very high.

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

Table 17: SCED cost driver.

<TODO: vedere cosa cambia se mettiamo high in SCED anche se il paramentro è sempre uguale.>
 <TODO: PAOLA che non è handicappata e GUIDO>

2.2.2.6 Final result <TODO: controllare l'ordine, mettere i parametri>

Cost Driver	Factor	Value
Required Software Reliability (RELY)	High	1.10
Data Base Size (DATA)	Very High	1.28
Product Complexity (CPLX)	High	1.17
Developed for Reusability (RUSE)	Nominal	1.00
Documentation Match to Life-Cycle Needs (DOCU)	Nominal	1.00
Execution Time Constraint (TIME)	High	1.11
Main Storge Constraint (STOR)	High	1.05
Platform Volatility (PVOL)	Low	0.87
Analyst Capability (ACAP)	High	0.85
Programmer Capability (PCAP)	High	0.88
Personnel Continuity(PCON)	Low	1.12
Applications Experience (APEX)	Low	1.10
Pltform Experience (PLEX)	Nominal	1.00
Language and Tool Experience (LTEX)	Low	1.09
Use of Software Tools(TOOL)	Very High	0.78
Multisite Development (SITE)	Extra High	0.80
Required Development Schedule (SCED)	Very High	1.00
Total		

Table 18: Final result of drivers evaluation.

3 Schedule

4 Resource Allocation

5 Risk Management

6 Appendices

6.1 Used Tools

- Microsoft Visio 2016: for all the diagrams in that document (as Testing Diagrams, etc ...);
- Lyx document processor: to write all the document;
- SourceTree: used as GitHub manager;
- GitHub: used to manage the shared building process of that document.

7 Hours of Work

Day	Guido Muscioni	Marco Orbelli	Paola Marchesini
17/01	1.5	1.5	1.5
18/01			
19/01			
20/01			
21/01			
22/01			
total			

List of Figures

1	New brand logo.	1
---	-------------------------	---

List of Tables

1	RELY cost driver.	9
2	DATA cost driver.	10
3	CPLX cost driver.	10
4	RUSE cost driver.	11
5	DOCU cost driver.	11
6	TIME cost driver.	11
7	STOR cost driver.	12
8	PVOL cost driver.	12
9	ACAP cost driver.	12
10	PCAP cost driver.	13
11	PCON cost driver.	13
12	APEX cost driver.	13
13	PLEX cost driver.	14
14	LTEX cost driver.	14
15	TOOL cost driver.	14
16	SITE cost driver.	15
17	SCED cost driver.	15
18	Final result of drivers evaluation.	16