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

1.1 Purpose

This document is base on all the previous document, its aim is to define the complete schedule of the project.
The document is divided in four section:

- Project Size, Cost and Effort Estimation: this section contains all the detailed information about the cost and effort. The estimation of the parameter is expressed with numeric value in order to retrieve a more realistic analysis;
- Schedule: this section contains an estimation of the general project progress;
- Resource Allocation: this section is based on the authors of all the document. Its aim is to show the schedule of all of them during the progress of the project;
- Risk Management: the aim of this section is to identify all the possible risks that can occur during the life of the system.

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

- **DD**: Design Document.
- **ITP**: Integration Test Plan.

- **ITPD**: Integration Test Plan Document.
- **DB**: Database.
- **RASD**: Requirement Analysis and Specification Document.
- **SDK**: Software Development Kit.
- **PhoneGap**: mobile cross-platform development framework.
- **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.
- **GeoJSON:** is a format for encoding a variety of geographic data structures use by Google Map API.
- **Payment Data Transfer** (**PDT**): is a notification service that, once activated, can send transaction-related information immediately to merchants who are using PayPal (PayPal Payments Standard).
- **COCOMO II**: is a mathematical model build in order to give an estimation of the cost and the effort needed to develop the entire system;
- **COCOMO II document**: is a reference to the model definition manual, that is reported in section1.3.

1.3 List of Reference Document

- Specification document: Assignments AA 2016-2017;
- Examples documents:
  - Project planning example document.
- Model Definition Manual by COCOMO II;
- `mondomatica.it/function.html`

2 Project Size, Cost and Effort Estimation

In this section there is a complete description of all the functionalities of PowerEnjoy system with respect to the level of complexity of them.
Section Size estimation: Function Points discloses the size estimation of the project. We have decided to base that analysis on the Function Points approach. That estimation does not take into account the user interface and the web tier developing. In actual fact an analysis of these two last things will be to rough, due to the variety of language used and the connection with the personal style of the developers.
Section Cost and Effort Estimation: COCOMO II presents the cost and effort estimation. We have decided to use COCOMO II to perform the analysis and retrieve the amount of time need in order to build and developed the whole system.

2.1 Size estimation: Function Points

This section is built using the Function Points approach to evalute 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 evalute 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.

EI	**Data Elements**		
Record Elements	**1-4**	**5-15**	**16+**
0-1	Low	Low	Average
2	Low	Average	Hight
3-4 +	Average	Hight	Hight

Table 1: EI values table.

EO / EQ	**Data Elements**		
Record Elements	**1-5**	**6-19**	**20+**
0-1	Low	Low	Average
2-3	Low	Average	Hight
4 +	Average	Hight	Hight

Table 2: EO/EQ values table.

ILF / ELF	Data Elements		
Record Elements	**1-19**	**20-50**	**51+**
1	Low	Low	Average
2-5	Low	Average	Hight
6+	Average	Hight	Hight

Table 3: ILF/ELF values table.

	Complexity Weight		
Function type	**Low**	**Average**	**Hight**
Internal Logic Files	7	10	15
External Logic Files	5	7	10
External Inputs	3	4	6
External Inquiries	3	4	6
External Outputs	4	5	7

Table 4: Overall values of Function Points.

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. According to what is mentioned in this bullet the level of complexity is set to average;

- 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. Instead, the second table contains the 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 all its phones for the employees, 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. Thus the level of complexity is low;

- Historical work data: the data about the "story" of employees' works are stored in a single table with an ID as primary key, where it can be found detailed information of a work: date, time, car involved,the couple of employees that have done the work, a brief description and a flag in order to understand if the phase of the car involved is changed. That (use of a flag) because an employee could remove a car from his/her list due to force majeur, leaving the work undone. So historical work data files have an average complexity;

- List data: the list data are very important for the system. In this table are stored the necessary information in order to identify a list of a cars: id of the list and the two employees who has to 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 to found all the members of the list (the specifications of that type of query can be found in the DD document). According to what is mentioned in this bullet the level of complexity is set to high;

- Car data: this internal files contain all the information about the cars which are managed by the PowerEnjoy system: code (primary key) used by the users to open/close the cars, license plate, battery level, number of seats and the information to identify the position of the cars (longitude and latitude). The information about the phase of the cars (that determinate the car state) are stored in a different table, containing the id of the car and its actual phase. The use of two tables simplify the management of the queries that only deal with the phase of the car. What is mentioined in this bullet lead us to set the complexity level to low;

- 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 users who have done the reservation or the ride , it is necessary to execute queries that involved other data structures. So that the level is set to hight;

- Ride data: the ride data is build with on a two-level structure. In the first table there is the general information to identify a ride: id, starting longitude, latitude, final longitude, 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). The level of complexity is low;

- Area data: this data structure contains two tables in order to distinguish the different aresa that are managed by the PowerEnjoy system. The special areas table contains the followings attribute: latitude, longitude, both uniquely identified the center, the radius and the number of plugs available. The second table contains information about the safe area external point: longitude, latitude and an order attribute, needed for the intepolation algorithm. The complexity of that files is set to low, in actual facts the difficult problem is the interpolation algorithms.

ILF	**Complexity**	**FPs**
User data	Average	10
Employee data	Low	7
Historical work data	Average	10
List data	High	15
Car data	Low	7
Reservation data	High	15
Ride data	Low	7
Area data	Low	7
Total		78

Table 5: Overall FPs for ILFs section.

(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 trough 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.

ELF	**Complexity**	**FPs**
	Google Map service	
Build the path	High	10
Create the map	High	10
	SMS service	
Read SMS	Average	7
	Payment service	
Save the payment	High	10
Aknowledge	High	10
Total		47

Table 6: Overall FPs for ELFs section.

2.1.3 External Inputs

The PowerEnjoy system provides a great number of functionalities which support interactions between people, the application or the web site. In this section the various interactions are explained and grouped by types of users:

General user:

- Sign in: this operation is simple because it involved only the account manager, which has to verify the login credentials used to sign in. It yelds 3 FPs;

- Sign up: this operation has an average complexity because the system has to verify all the credentials inserted by the user and especially the payment information. It yelds 4 FPs.

User verified:

- Look for car nearby: this operation has an hight level of complexity. It is based on an algorithm that queries the DB, asking for nearby car, using the position of the user as a parameter. It also has to managed a great number o attributes for all the cars. It yelds 6 FPs;
- Insert postition: this is a simple operation, the user inserts manually his/her position. It yelds 3 FPs;
- Select the car: thanks to the look for car nearby operation, the user has already all the information needed to select a car. It yelds 3 FPs;
- Reserve that car: this operation has a hight complexity. It involves a lot of components which have to communicate, to query and to update the DB. It also sends notifications to the users. It yelds 6 FPs;
- Cancel the reservation: this is an operation of an average complexity. It calls a controller which has to update the DB, and then, with a notify to the car controller, the table of cars are also updated. It yelds 4 FPs;
- See the way to the car: this operation has an hight complexity. It requires the interaction with the map gateway to generate the path to the car. The information returned from the gateway need also a medium computation. It yelds 6 FPs;
- Open the car: this operation has a high level of complexity. That assertion is based on the possible problems that this operation can create, as a malfunctioning in the car sensor. So we decided to give 6 FPs;
- Modify: this operation has an aveage complexity (according to the "Sign up" operation) because the system has to verify the new credentials inserted by the user. It yelds 4 FPs.

Employee:

- Look for my list: this operation has a hight complexity: it needs a complex query on the DB, looking for the cars that require attention. The list contains also all the cars information It yelds 6 FPs;
- See the state of the car: thanks to the look for my list operation, the employee has already all the information needed to select a car. It yelds 3 FPs;
- See car in a map: this operation has a high level of complexity. It has to call the map gateway in order to draw the map with the car placed in their current positions. It yelds 6 FPs;
- Change state of car: this operation has an hight complexity. When a work is finisched, the employee has to change the phase of the car. This functionality includes both queries on the DB and interactions between the controllers. It also requires to remove the car and consequently updates the DB. It yelds 6 FPs.

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 yelds 3 FPs;
- Activate money saving option: this operation has an hight complexity. It requires queries on the DB to show the Special Area nearby the destination of the user. It also include the computation of the free plugs. Finally that operation has to update the discount that must to be applied at the end of the ride. It yelds 6 FPs;
- Call / Terminate the call: this is a straighforward operation, it simply calls the service center, thanks to the mobile device included in the car system. It yelds 3 FPs;
- Finish the ride: this operation is an average operation. In actual fact this functionality updates only a table of the DB, modifying the state of the ride. It yelds 4 FPs.

EI	Complexity	FPs
General user		
Sign in	Low	3
Sign up	Low	3
User verified		
Look for car nearby	High	6
Insert position	Low	3
Select the car	Low	3
Reserve that car	High	6
Cancel the reservation	Average	4
See the way to the car	High	6
Open the car	High	6
Modify	Average	4
Employee		
Look for my list	High	6
See the state of the car	Low	3
See car in a map	High	6
Change state of car	High	6
Car driver		
Skip that page	Low	3
Activate money saving option	High	6
Call/Terminate the call	Low	2×3
Finish the ride	Average	4
Total		85

Table 7: Overall FPs for EI section.

2.1.4 External Inquiries

In PowerEnjoy system there are only two pure request of data that the user could do. One is made by the user, and the other by the employee.

User:

- Retrieve account information: this is a simple operation. In actual fact the user request all his/her information. It yelds 3 FPs;

Employee:

- See the list of car: that operation is made by the employee in order to see the list of the car that he/she has to take care, once the list has been generated. It yelds 3 FPs;

EI	Complexity	FPs
User		
Retriev account information	Low	3
Employee		
See the list of car	Low	3
Total		6

Table 8: Overall FPs for EQ section.

2.1.5 External Outputs

The PowerEnjoy system needs also to interact directly with the user, sending him/her a mail or a SMS in order 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 sent to the user through the reservation message.

EI	Complexity	FPs
Notification for credentials	Low	4
Mail for the bill	Low	4
SMS notification	Low	4
Total		12

Table 9: Overall FPs for EO section.

2.1.6 Overall Estimation

The following table show the results of our extimation:

Lower Bound:

$$SLOC = 228 \times 15 = 3420$$

Function Type	Value
Internal Logic Files	78
External Logic Files	47
External Inputs	85
External Inqueries	6
External output	12
Total	228

Table 10: Estimation in FPs.

Average value:

$$SLOC = 228 \times 46 = 10488$$

Upper Bound:

$$SLOC = 228 \times 67 = 15276$$

2.2 Cost and Effort Estimation: COCOMO II

The COCOMO approach is based on assign a level of complexity for each factors and drivers defined by the mathematical model. The tables reported in that section are all taken from the documents reported in section List of Reference Document(1.3).

2.2.1 Scale Factors

According to the model definition manual of COCOMO II these values are used in order to determined the effect of economies or diseconomies of scale. The result of that analysis give to us a parameter that represents the relation between the productivity and the product size.

- **PREC**: we have never implemented a project with that size, so the value is set to very low;
- **FLEX**: we have to fulfill al the requirements gives to use from the stakeholders, so the level is set to rigorous;
- **RESL**: according to section Risk Management we think that we have covered as many risks as possible. So the level is set to mostly;
- **TEAM**: we are very cooperative ans cohesive, so the level is set to seamless interaction;
- **PMAT**: according to the description in COCOMO II manual, the level is set to nomina.

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC **SF_j**	**Thoroughly unprecedented** **6.20**	Largely unprecedented 4.96	Somewhat unprecedented 3.72	Generally familiar 2.48	Largely familiar 1.24	Thoroughly familiar 0.00
FLEX **SF_j**	**Rigorous** **5.07**	Occasional relaxation 4.05	Some relaxation 3.04	General conformity 2.03	Some conformity 1.01	General goals 0.00
RESL **SF_j**	Little (20%) 7.07	Some (40%) 5.65	Often (60%) 4.24	Generally (75%) 2.83	**Mostly (90%)** **1.41**	Full (100%) 0.00
TEAM **SF_j**	Very difficult interactions 5.84	Some difficult interactions 4.38	Basically cooperative interactions 3.29	Largely cooperative 2.19	Highly cooperative 1.10	**Seamless interaction** **0.00**
PMAT	The Estimated Equivalent Process Maturity Level (EPML) or					
SF_j	SW-CMM Level 1 Lower 7.80	SW-CMM Level 1 Upper 6.24	**SW-CME Level 2** **4.68**	SW-CMM Level 3 3.12	SW-CMM Level 4 1.56	SW-CMM Level 5 0.00

Table 11: TIME scale factors diagram.

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 12: RELY cost driver.

DataBase Size (DATA) With a quick analysis of the DB tables defined in DD document, we think that a 5 GB database is enough in order to store ten years of business. According to that, the D/P is equal to 196e3, so the level of that driver is set to very high.

DATA Descriptors:		Testing DB bytes/Pgm SLOC < 10	$10 \leq \frac{D}{P} < 100$	$100 \leq \frac{D}{P} < 100$	$\mathbf{\frac{D}{P} \geq 1000}$	
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 13: 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 14: 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 15: 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 mantain 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 16: 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 17: TIME cost driver.

Main Storage Constraint (STOR) That value represents the percentage of space that the system is going to use for its correct behaviour. As we would not waste a lot of resorces 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 18: 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 19: 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 levl 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 20: 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 21: 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 22: 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:	$\leq$ 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 23: 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 school level because we have used that platforms only during some university projects. According to that we decided to set PLEX to nominal.

PLEX Descriptors:	$\leq$ 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 24: 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 databse; so the level is set to low.

LTEX Descriptors:	$\leq$ 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 25: 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 26: 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 communica-tion	Wideband elect. comm., occasional video conf	**Interactive multimedi**
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 27: 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 28: SCED cost driver.

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)	Nominal	1.00
Total		1.23175

Table 29: Final result of drivers evaluation.

2.2.2.6 Final result

$$PM = A \times Size^E \prod_{i=1}^{17} EM_i$$

With our parameters:

Lower Bound:

$$PM = 20.785$$

Average value:

$$PM = 70.002$$

Upper Bound:

$$PM = 105.216$$

So the effort can be estimated with that formula:

$$Schedule(TDEV) = [C \times (PM_{NS})^{(D+0.2\times(E-B))}] \times \frac{SCED\%}{100}$$

$$(D + 0.2 \times (E - B)) = 0.3147$$

The three values that emerge from our analysis are:

Lower Bound:

$$Schedule = 9.54 months$$

Average value:

$$Schedule = 13.97 months$$

Upper Bound:

$$Schedule = 15.89 months$$

3 Schedule

The project schedule management plan is a component of the project plan document that establishes the activities for developing, monitoring, and controlling the project
So in this section we're going to provide a general, high-level schedule giving information about the development progress of the system. The aim of this part is to establish a general baseline of the project line up. More refined schedules will be defined during the course of the project, in order to manage the internal organization of the single development phases. The diagrams reported below are not completely realistic; in actual fact we are not going to develop the project. So that is only an ipothetical view of the time schedule of the development phases. However, according to the previous analysis that we have made in that document we have tried to do these schedule as more realistica s possible. We have hypothesized that the whole realization of the project covering a period from October 2016 to November 2017.

ID	Nome attività	Inizio	Fine	Durata
1	**RASD**	**14/10/2016**	**15/11/2016**	**33g**
2	Identfication of stakeholders	14/10/2016	15/10/2016	2g
3	Identfication of actors	14/10/2016	16/10/2016	3g
4	Definition of system requirements	18/10/2016	21/10/2016	4g
5	Definition of assumptions	20/10/2016	23/10/2016	4g
6	Definition of constraints	24/10/2016	24/10/2016	1g
7	User interface design (MockUp)	25/10/2016	31/10/2016	7g
8	Realization of use case diagrams	26/10/2016	26/10/2016	1g
9	Realization of sequence diagram	27/10/2016	29/10/2016	3g
10	Realization of activity diagrams	31/10/2016	31/10/2016	1g
11	Realization of state chart diagrams	31/10/2016	31/10/2016	1g
12	Realization of class diagram	01/11/2016	02/11/2016	2g
13	Initial draft of alloy	04/11/2016	06/11/2016	3g
14	Final alloy	07/11/2016	10/11/2016	4g
15	Meeting with stakelhoders	11/11/2016	11/11/2016	1g
16	Final refinements	14/11/2016	15/11/2016	2g
17	**DD**	**16/11/2016**	**15/12/2016**	**30g**
18	Definitions of architectural styles and patterns	16/11/2016	18/11/2016	3g
19	Realization of component view	21/11/2016	22/11/2016	2g
20	Realization of deployment view	22/11/2016	23/11/2016	2g
21	Realization of runtime view	25/11/2016	29/11/2016	5g
22	Development of component interface	01/12/2016	02/12/2016	2g
23	Overview of algorithms design	05/12/2016	07/12/2016	3g
24	Realization of UX diagrams	08/12/2016	11/12/2016	4g
25	Realization of BCE diagrams	12/12/2016	15/12/2016	4g
26	Meeting with stakelhoders	14/12/2016	14/12/2016	1g
27	Final refinements	15/12/2016	15/12/2016	1g
28	**Development**	**16/12/2016**	**13/06/2017**	**180g**
29	Subcomponent development	16/12/2016	05/03/2017	80g
30	External component acquisition and analysis	06/02/2017	05/03/2017	28g
31	Component development	07/03/2017	15/04/2017	40g
32	Component integration	17/04/2017	31/05/2017	45g
33	Refinements	01/06/2017	13/06/2017	13g
34	**Verification and validation**	**14/06/2017**	**22/08/2017**	**70g**
35	Junit test	14/06/2017	13/07/2017	30g
36	Code inspection	14/06/2017	03/07/2017	20g
37	Definition of drivers and stubs	21/06/2017	05/07/2017	15g
38	Integration testing	03/07/2017	08/08/2017	37g
39	Final testing	07/08/2017	16/08/2017	10g
40	Meeting with stakelhoders	17/08/2017	17/08/2017	1g
41	Final refinements	18/08/2017	22/08/2017	5g
42	**Deployement**	**24/08/2017**	**02/10/2017**	**40g**
43	System deployement	24/08/2017	10/09/2017	18g
44	Teaching meeting with administration personnel	11/09/2017	15/09/2017	5g
45	App distribution	18/09/2017	19/09/2017	2g
46	Staff formation meeting	18/09/2017	27/09/2017	10g
47	Final preparation	28/09/2017	02/10/2017	5g
48	**System start up**	**04/10/2017**	**12/11/2017**	**40g**
49	System first start	04/10/2017	05/10/2017	2g
50	Real-life simulation	06/10/2017	20/10/2017	15g
51	Real world test and bugs fix	23/10/2017	06/11/2017	15g
52	Main updates and checks	03/11/2017	12/11/2017	10g

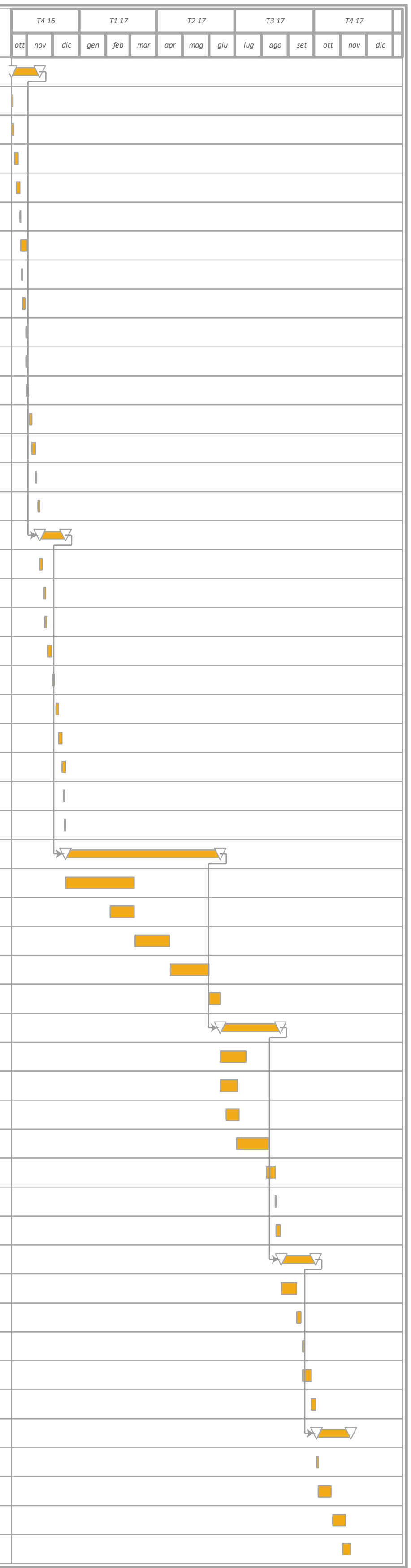

Figure 2: General Gantt diagram.

4 Resource Allocation

The project resource allocation management plan is a component of the project plan document that establishes which activities are carried out by the PowerEnjoy developer.
So in this section we're going to provide a general overview of how the tasks defined by the schedule in the previous section will be fulfilll between the three members of the PowerEnjoy development team. More refined schedules will be defined during the project in order to manage the internal organization of the single development phases. As we already mentioned in the previous section the diagrams reported below are not completely realistic, in fact we have also included activities in the requirement analysis, design phases that won't actually take place, like the stakeholders meetings, as well as the full implementation phase. This has been done to have achieve a more realistic depiction of how the development process could go.

ID	Nome attività	Inizio	Fine	Durata
1	**RASD**	**14/10/2016**	**15/11/2016**	**33g**
2	Identfication of stakeholders	14/10/2016	15/10/2016	2g
3	Identfication of actors	14/10/2016	16/10/2016	3g
4	Definition of system requirements	18/10/2016	21/10/2016	4g
5	Definition of assumptions	20/10/2016	23/10/2016	4g
6	Definition of constraints	24/10/2016	24/10/2016	1g
7	User interface design (MockUp)	25/10/2016	31/10/2016	7g
8	Realization of use case diagrams	26/10/2016	26/10/2016	1g
9	Realization of sequence diagram	27/10/2016	29/10/2016	3g
10	Realization of activity diagrams	31/10/2016	31/10/2016	1g
11	Realization of state chart diagrams	31/10/2016	31/10/2016	1g
12	Realization of class diagram	01/11/2016	02/11/2016	2g
13	Initial draft of alloy	04/11/2016	06/11/2016	3g
14	Final alloy	07/11/2016	10/11/2016	4g
15	Meeting with stakelhoders	11/11/2016	11/11/2016	1g
16	Final refinements	14/11/2016	15/11/2016	2g
17	**DD**	**16/11/2016**	**15/12/2016**	**30g**
18	Definitions of architectural styles and patterns	16/11/2016	18/11/2016	3g
19	Realization of component view	21/11/2016	22/11/2016	2g
20	Realization of deployment view	22/11/2016	23/11/2016	2g
21	Realization of runtime view	25/11/2016	29/11/2016	5g
22	Development of component interface	01/12/2016	02/12/2016	2g
23	Overview of algorithms design	05/12/2016	07/12/2016	3g
24	Realization of UX diagrams	08/12/2016	11/12/2016	4g
25	Realization of BCE diagrams	12/12/2016	15/12/2016	4g
26	Meeting with stakelhoders	14/12/2016	14/12/2016	1g
27	Final refinements	15/12/2016	15/12/2016	1g
28	**Development**	**16/12/2016**	**13/06/2017**	**180g**
29	Subcomponent development	16/12/2016	05/03/2017	80g
30	External component acquisition and analysis	06/02/2017	05/03/2017	28g
31	Component development	07/03/2017	15/04/2017	40g
32	Component integration	17/04/2017	31/05/2017	45g
33	Refinements	01/06/2017	13/06/2017	13g
34	**Verification and validation**	**14/06/2017**	**22/08/2017**	**70g**
35	Junit test	14/06/2017	13/07/2017	30g
36	Code inspection	14/06/2017	03/07/2017	20g
37	Definition of drivers and stubs	21/06/2017	05/07/2017	15g
38	Integration testing	03/07/2017	08/08/2017	37g
39	Final testing	07/08/2017	16/08/2017	10g
40	Meeting with stakelhoders	17/08/2017	17/08/2017	1g
41	Final refinements	18/08/2017	22/08/2017	5g
42	**Deployement**	**24/08/2017**	**02/10/2017**	**40g**
43	System deployement	24/08/2017	10/09/2017	18g
44	Teaching meeting with administration personnel	11/09/2017	15/09/2017	5g
45	App distribution	18/09/2017	19/09/2017	2g
46	Staff formation meeting	18/09/2017	27/09/2017	10g
47	Final preparation	28/09/2017	02/10/2017	5g
48	**System start up**	**04/10/2017**	**12/11/2017**	**40g**
49	System first start	04/10/2017	05/10/2017	2g
50	Real-life simulation	06/10/2017	20/10/2017	15g
51	Real world test and bugs fix	23/10/2017	06/11/2017	15g
52	Main updates and checks	03/11/2017	12/11/2017	10g

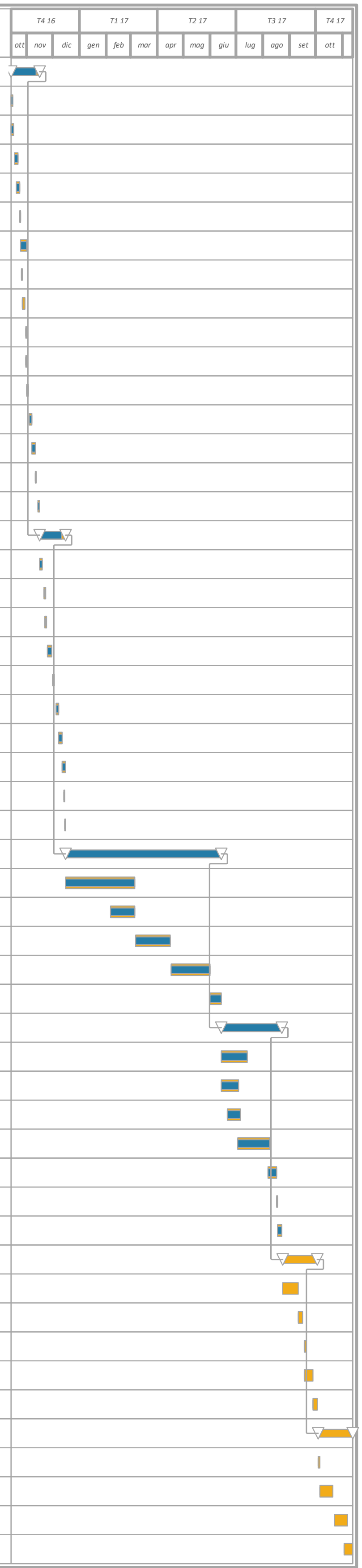

Figure 3: Muscioni Guido Gantt diagram.

ID	Nome attività	Inizio	Fine	Durata
1	**RASD**	**14/10/2016**	**15/11/2016**	**33g**
2	Identfication of stakeholders	14/10/2016	15/10/2016	2g
3	Identfication of actors	14/10/2016	16/10/2016	3g
4	Definition of system requirements	18/10/2016	21/10/2016	4g
5	Definition of assumptions	20/10/2016	23/10/2016	4g
6	Definition of constraints	24/10/2016	24/10/2016	1g
7	User interface design (MockUp)	25/10/2016	31/10/2016	7g
8	Realization of use case diagrams	26/10/2016	26/10/2016	1g
9	Realization of sequence diagram	27/10/2016	29/10/2016	3g
10	Realization of activity diagrams	31/10/2016	31/10/2016	1g
11	Realization of state chart diagrams	31/10/2016	31/10/2016	1g
12	Realization of class diagram	01/11/2016	02/11/2016	2g
13	Initial draft of alloy	04/11/2016	06/11/2016	3g
14	Final alloy	07/11/2016	10/11/2016	4g
15	Meeting with stakelhoders	11/11/2016	11/11/2016	1g
16	Final refinements	14/11/2016	15/11/2016	2g
17	**DD**	**16/11/2016**	**15/12/2016**	**30g**
18	Definitions of architectural styles and patterns	16/11/2016	18/11/2016	3g
19	Realization of component view	21/11/2016	22/11/2016	2g
20	Realization of deployment view	22/11/2016	23/11/2016	2g
21	Realization of runtime view	25/11/2016	29/11/2016	5g
22	Development of component interface	01/12/2016	02/12/2016	2g
23	Overview of algorithms design	05/12/2016	07/12/2016	3g
24	Realization of UX diagrams	08/12/2016	11/12/2016	4g
25	Realization of BCE diagrams	12/12/2016	15/12/2016	4g
26	Meeting with stakelhoders	14/12/2016	14/12/2016	1g
27	Final refinements	15/12/2016	15/12/2016	1g
28	**Development**	**16/12/2016**	**13/06/2017**	**180g**
29	Subcomponent development	16/12/2016	05/03/2017	80g
30	External component acquisition and analysis	06/02/2017	05/03/2017	28g
31	Component development	07/03/2017	15/04/2017	40g
32	Component integration	17/04/2017	31/05/2017	45g
33	Refinements	01/06/2017	13/06/2017	13g
34	**Verification and validation**	**14/06/2017**	**22/08/2017**	**70g**
35	Junit test	14/06/2017	13/07/2017	30g
36	Code inspection	14/06/2017	03/07/2017	20g
37	Definition of drivers and stubs	21/06/2017	05/07/2017	15g
38	Integration testing	03/07/2017	08/08/2017	37g
39	Final testing	07/08/2017	16/08/2017	10g
40	Meeting with stakelhoders	17/08/2017	17/08/2017	1g
41	Final refinements	18/08/2017	22/08/2017	5g
42	**Deployement**	**24/08/2017**	**02/10/2017**	**40g**
43	System deployement	24/08/2017	10/09/2017	18g
44	Teaching meeting with administration personnel	11/09/2017	15/09/2017	5g
45	App distribution	18/09/2017	19/09/2017	2g
46	Staff formation meeting	18/09/2017	27/09/2017	10g
47	Final preparation	28/09/2017	02/10/2017	5g
48	**System start up**	**04/10/2017**	**12/11/2017**	**40g**
49	System first start	04/10/2017	05/10/2017	2g
50	Real-life simulation	06/10/2017	20/10/2017	15g
51	Real world test and bugs fix	23/10/2017	06/11/2017	15g
52	Main updates and checks	03/11/2017	12/11/2017	10g

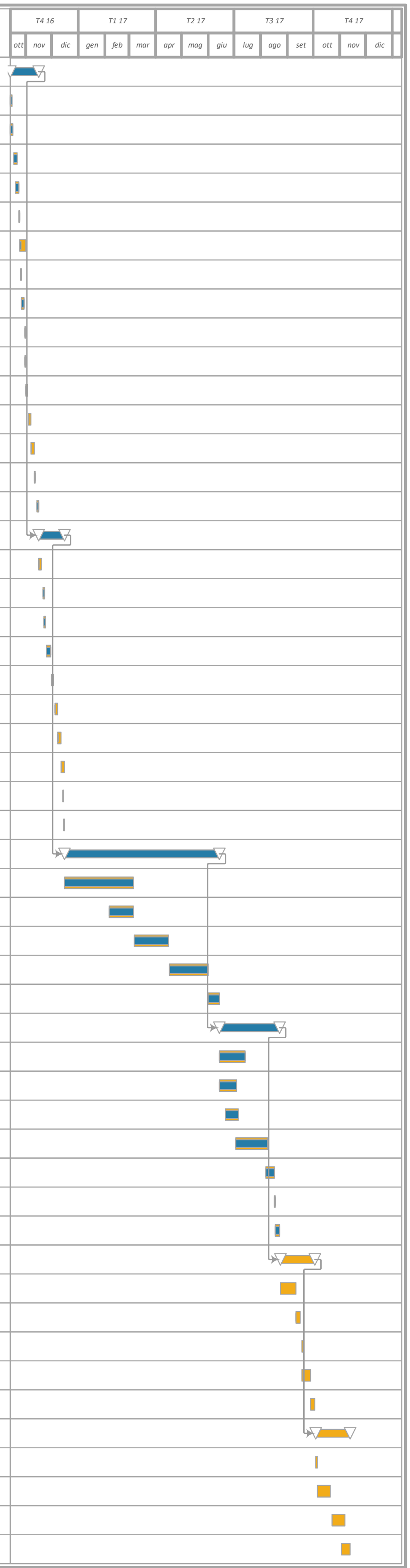

Figure 4: Orbelli Marco Gantt diagram.

ID	Nome attività	Inizio	Fine	Durata
1	**RASD**	**14/10/2016**	**15/11/2016**	**33g**
2	Identfication of stakeholders	14/10/2016	15/10/2016	2g
3	Identfication of actors	14/10/2016	16/10/2016	3g
4	Definition of system requirements	18/10/2016	21/10/2016	4g
5	Definition of assumptions	20/10/2016	23/10/2016	4g
6	Definition of constraints	24/10/2016	24/10/2016	1g
7	User interface design (MockUp)	25/10/2016	31/10/2016	7g
8	Realization of use case diagrams	26/10/2016	26/10/2016	1g
9	Realization of sequence diagram	27/10/2016	29/10/2016	3g
10	Realization of activity diagrams	31/10/2016	31/10/2016	1g
11	Realization of state chart diagrams	31/10/2016	31/10/2016	1g
12	Realization of class diagram	01/11/2016	02/11/2016	2g
13	Initial draft of alloy	04/11/2016	06/11/2016	3g
14	Final alloy	07/11/2016	10/11/2016	4g
15	Meeting with stakelhoders	11/11/2016	11/11/2016	1g
16	Final refinements	14/11/2016	15/11/2016	2g
17	**DD**	**16/11/2016**	**15/12/2016**	**30g**
18	Definitions of architectural styles and patterns	16/11/2016	18/11/2016	3g
19	Realization of component view	21/11/2016	22/11/2016	2g
20	Realization of deployment view	22/11/2016	23/11/2016	2g
21	Realization of runtime view	25/11/2016	29/11/2016	5g
22	Development of component interface	01/12/2016	02/12/2016	2g
23	Overview of algorithms design	05/12/2016	07/12/2016	3g
24	Realization of UX diagrams	08/12/2016	11/12/2016	4g
25	Realization of BCE diagrams	12/12/2016	15/12/2016	4g
26	Meeting with stakelhoders	14/12/2016	14/12/2016	1g
27	Final refinements	15/12/2016	15/12/2016	1g
28	**Development**	**16/12/2016**	**13/06/2017**	**180g**
29	Subcomponent development	16/12/2016	05/03/2017	80g
30	External component acquisition and analysis	06/02/2017	05/03/2017	28g
31	Component development	07/03/2017	15/04/2017	40g
32	Component integration	17/04/2017	31/05/2017	45g
33	Refinements	01/06/2017	13/06/2017	13g
34	**Verification and validation**	**14/06/2017**	**22/08/2017**	**70g**
35	Junit test	14/06/2017	13/07/2017	30g
36	Code inspection	14/06/2017	03/07/2017	20g
37	Definition of drivers and stubs	21/06/2017	05/07/2017	15g
38	Integration testing	03/07/2017	08/08/2017	37g
39	Final testing	07/08/2017	16/08/2017	10g
40	Meeting with stakelhoders	17/08/2017	17/08/2017	1g
41	Final refinements	18/08/2017	22/08/2017	5g
42	**Deployement**	**24/08/2017**	**02/10/2017**	**40g**
43	System deployement	24/08/2017	10/09/2017	18g
44	Teaching meeting with administration personnel	11/09/2017	15/09/2017	5g
45	App distribution	18/09/2017	19/09/2017	2g
46	Staff formation meeting	18/09/2017	27/09/2017	10g
47	Final preparation	28/09/2017	02/10/2017	5g
48	**System start up**	**04/10/2017**	**12/11/2017**	**40g**
49	System first start	04/10/2017	05/10/2017	2g
50	Real-life simulation	06/10/2017	20/10/2017	15g
51	Real world test and bugs fix	23/10/2017	06/11/2017	15g
52	Main updates and checks	03/11/2017	12/11/2017	10g

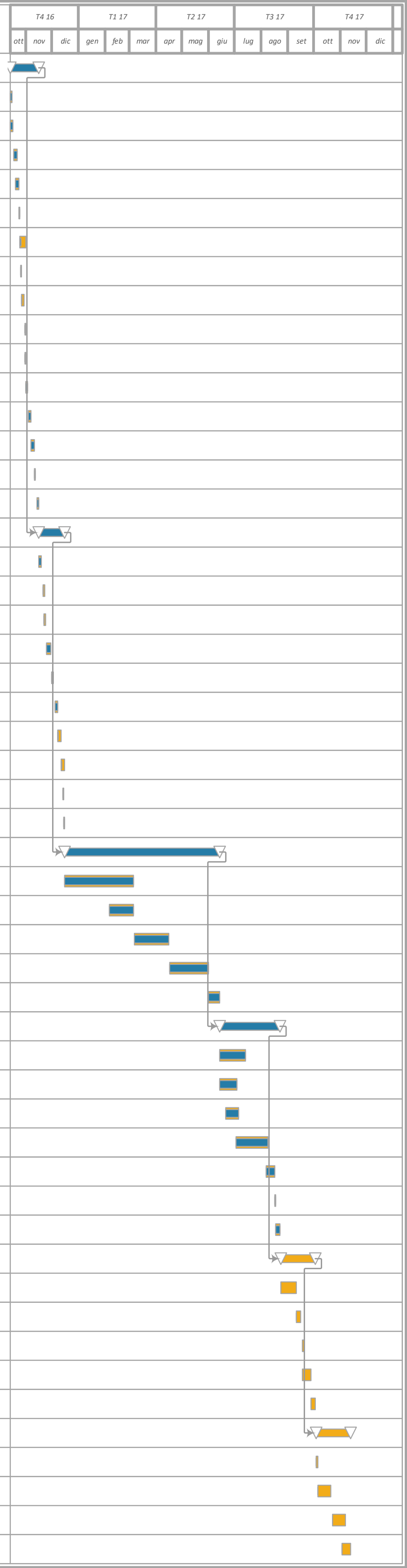

Figure 5: Marchesini Paola Gantt diagram.

5 Risk Management

Risk management is a key phase of our project.
So the developers, and all other people that work on the project, can have a complete view of the risks that may affect the system. We assume that a risk is a potential problem, which might or not might happen, that can result in a technical malfunction or human difficulties.
In the section presented above there are all the risks that we have indentified during the building of this and all the previous documents.

5.1 Project Risks

In this section there are all the risks that threaten the project plan. That kind of problems can flow into an unexpected extension of the schedule define in section 3, and clearly in a cost increasing.

Personnel shortfall
That risk emerge from the loss of one or more members of the project team.
As we know from the start that a problem like that could happen in every moments; we always work as a strong and cooperative team, as we mentioned in the cost drivers of COCOMO II. So that the remaining people in the team can easely substitute the missing members. Obviously the schedule will slip, but will not exceed the upper bound TDEV defined in section 2.

Unrealistic schedules
That risk involves errors during the schedule phase of the project. In particular, it can emerge from a strict deadline from the stakeholders.
We have tried to take that risk into account by make a more flexible time schedule estimation. In actual fact the variace between the lower and the upper bound of schedule, defined trough COCOMO II, is high.

5.2 Technical Risks

Inexperience team members
That risk emerges during the developing phase of the project. It can be produced by two main reasons:

- The lack of specialized skills required in order to achieve the final version of the system;
- An overstimation of the knowledge of one or more team members.

We have already taken that problem into account during the cost and effort analysis. The parameters that deal with what we have mentioned in this paragraph have been understimated. So that the schedule of the project will not change due to this risk, it will remain in the range of time that we have already defined.
In addiction, we have always used tools and languages that we have already known from our university studies, so that risk is well coverage.

Bad external components
That risk emerges from the use of external components that have not been developed well.
During the building of that project we have always selected well known system. So that the probability that a problem occur in that case is remote. Moreover the external systems used in our project are develop by multinational companies that can resolve that kind of problems in a very efficient way.

Real-Time performance shortfalls
That risk emerges when the system is realesed in the world.
Firstly, we have made a very efficient test plan document, that can cover all the main functionalities of the system.
Secondly if a problem like that occurs, and result in a malfunctioning in a server of our arrchitecture; the two servers farms designed will make that problem not recognizable by the users. In actual fact, as it was mentioned in the DD document if a server goes down the other can replace all the functionalities of the missing one.

Hacker attack
That risk emerges from smeone that tries to access to our system without permission.
The solution of that risk is already written in the previous paragraph: Real-Time performance shortfalls.

Wrong user interface
That problem can emerge from a malfunctioning or a wrong design in the user interface.
Even if we have designed it properly, if an error occurs, thank to the use of Phonegap, the bug fix must be done only in one code. The reason of that solution is well explained in the previous documents.

5.3 Business Risks

Gold plating

We have decided to put that risk in the business risks section even if it also deals with the possible technical risks.
That risk emerge from a maniacal approach to the develop of some part of the system.
Also in that case the big variance in the schedule lead us to identify and resolve that kind of risk.

Unrealistic budget
That risk can emerges from an understimation of the whole cost of the project.
As we have already mentioned, the estimation trough the COCOMO II and the FPs approach has been done very carefully. Furthermore, the bottom up approach used in order to develop the project implicate that is very unlikely that a huge expense can happen without any signs.

6 Appendices

6.1 Used Tools

- Microsoft Visio 2016: for all the diagrams in that document (as Testing Diagrams, etc ...);
- Microsoft Excel 2016: for the Gantt defined;
- 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	4	2	2
19/01	2	3	3
20/01	4	4	2
21/01	3	3	3
22/01	5	5	5
total	19.5	18.5	16.5

List of Figures

List of Tables