POLITECNICO
MILANO 1863

Project Plan Document

Joan Ficapal Vila (876805), Nicolò Vendramin (879113)

January 23, 2017
v1.0

Revision History

Revision	Date	Author(s)	Description
0.1	18.01.17	N and J	Initiated the document
0.2	18.01.17	N and J	Writing the introduction
0.3	19.01.17	N and J	Work to define the outline to be followed for Function Points and Cocomo
0.4	19.01.17	N	Starting with function points evaluation
0.5	19.01.17	J	Starting with COCOMO analysis
0.6	20.01.17	N and J	Meeting for common revision
0.7	20.01.17	N	Concluded work on Function points
0.8	21.01.17	J	Concluded work on COCOMO
0.9	21.01.17	N and J	Schedule planning
0.10	21.01.17	N	Risk Management
0.11	22.01.17	J	Resource Management
0.12	22.01.17	N and J	Revision
1.0	22.01.17	N and J	First Release
1.1	23.01.17	N and J	Small fixes

Hours of work

- Joan Ficapal Vila : 20 hours
- Nicolò Vendramin : 20 hours

Contents

1 Introduction

1.1 Purpose and Scope

This document is the Project Plan Document for the PowerEnjoy application. The main purpose of the following document is to estimate the complexity of the project in order to assist the project leader in the activity of estimations of costs and efforts. The first part of the document will be dedicated to the estimation of the size of the project using the function points method, together with a cost and effort estimation done following the COCOMO approach.
In the second part of the document we are going to propose a possible schedule for the activities that compose the project, from the elicitation and identification of the requirements to the implementation and testing activities. After that the document will present a possible scheme for the allocation of the human resources available for the project to the different activities composing the schedule, and at last an analysis concerning the possible risks that project could face in the different stages of develpment.

1.2 Definitions, Acronyms and Abbreviations

1.2.1 Acronyms

- **FP:** Funtion Points. Is a method to estimate the dimension of the source code needed to implement the functionalities of a program.
- **SLOC:** Source Lines of Code. Lines of source code. The number of line needed for the implementation of a program.
- **API:** Access Point of Interface. APIs are the methods exposed by a software product to be used by other softwares.
- **ELF:** External Logical File. One of the function types used in the Function Points method.
- **ILF:** Internal Logical File. One of the function types used in the Function Points method.
- **EI:** External Input. One of the function types used in the Function Points method.
- **EO:** External Output. One of the function types used in the Function Points method.
- **EQ:** External inQuiries. One of the function types used in the Function Points method.
- **COCOMO II:** COnstructive COst MOdel. Is a model that allows to estimate the cost, effort and schedule when planning a new software development activity.

1.2.2 Definitions

- **Unit Testing:** With unit testing is meant that activity that make sure that every basic unit that composes the software is correctly working as expected. With unit we indicate the smaller atom of which the software is composed. In case of a Java program we can assume Classes as units.
- **Integration Testing:** With integration testing is meant that phase of software testing in which individual modules are tested together as a group.

1.3 Reference Documents

Here we attached the list of all the documents that are kept as references in the writing of this document.

- Templates provided at lesson.
- Slides of the course.
- COCOMO II official documentation.
- Function Points method documentation.

2 Project size, cost and effort estimation

2.1 Size estimation: function points

For the estimation of the project size we chose to use the method of the function points. This method has been defined in 1975 by Allan Albrecth and is based on the assumption the dimension of any software product can be estimated basing on the functionalities that it has to offer. In particular basing on the combinations of some particular software characteristics (Data structures, inputs and outputs, inquiries and external services) to whom are associated specific weights, is possible to estimate the dimension of the software to be developed. The steps that must be followed to apply this method are mainly the identification and classification of the different function points, followed by the weighted sum of them which is an estimator of the number of lines of code needed to implement the identified functional points. The number of functional points to assign to each element of a certain function type have been derived from the analysis of real project developed in the past and condensed in the following tables that we attach at the end of the paragraph.

	Data Elements		
Record Elements	*1-19*	*20-50*	*51+*
1	Low	Low	Avg
2-5	Low	Avg	High
6+	Avg	High	High

Figure 1: Internal Logic Files, Complexity Weight Assignment

	Data Elements		
File Types	*1-4*	*5-15*	*16+*
0-1	Low	Low	Avg
2-3	Low	Avg	High
4+	Avg	High	High

Figure 2: Exernal Input, Complexity Weight Assignment

	Data Elements		
File Types	*1-5*	*6-19*	*20+*
0-1	Low	Low	Avg
2-3	Low	Avg	High
4+	Avg	High	High

Figure 3: External Output and Inquiries, Complexity Weight Assignment

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

Figure 4: Complexity Weight translation to Function Points, for each Function Type

2.1.1 Internal Logic Files (ILFs)

Internal logic files is the first type of function points and it is the one that is that represents the set of data that are used and managed by the application. In the case of our application the data that are internally managed by the application are the ones stored in the internal database about: Clients, Areas, Cars, Service, Bill, Cookie Session.
Client are stored in a table, with an unique index that is used in a second table to associate every client id to the password. A third table stores informations about the areas (area id, reference adress, type of area). In another table the points that form the limits for the polygon that identifies every area are stored in association to the id of the correspondin area. An additional table is dedicated to the informations about the cars that form the system and in particular it contains the Plate, the location, the state of the car (Booked, Free or inRide), the boolean that represent wether the engine is on or off, the number of occupied seats and the status of the battery. Services are stored in two different tables depending on wether they are bookings or rides. In both cases the key is given by the combination of car id, client id and timestamp followed by the different attributes of the service itself (that vary depending on the type of service). The bills are stored in a table where to every service id is attributed the corresponding amount of money charged to the user and the timestamp of the invocation of the billing system. Cookie sessions are stored in a table composed of three attributes: cookie session, user id (optional), and timestamp. A special table is dedicated to store the booking timers: this table is made up of associations between service id and timestamps.

Internal Logic File	Complexity	FPs
Clients data	Average	10
Data about areas	Low	7
Data about vehicles	Low	7
Services data	Average	10
Data about bills	Low	7
Cookie session storage	Low	7
Booking timers storage	Low	7
	Total	55

Figure 5: Internal Logic Files, Resume Table

2.1.2 External Logic Files (ELFs)

The second functional type is External Interface Files amd represents all those source of data external to the application its users. In the PowerEnjoy application the impact of External Interface Files is quite important since the main source of data for the system is the existing systems that handles the data concerning the operation of the application. The main interactions are related to the vehicles rented by the service. The information about them is not queried by our system directly, instead is updated by the existing system that after every relevant change updates the state of the private database of the PowerEnjoy application. Some operations in the interaction with this system return data that must be processed and used by the application: in particular those interactions are the one in which the Login, Register, ShowAccountInfo, ShowCarDetails and ShowVehicles apis of the existing system. Since the elaboration of the returned data has a low complexity we assign to each of this external interface files 10 FPs, so that summing them up we obtain a total of 50 FPs.

External Logic File	Complexity	FPs
Login data check	Low	10
Register data	Low	10
Gather account info	Low	10
Gather car details	Low	10
Get vehicles in zone	Low	10
	Total	50

Figure 6: External Logic Files, Resume Table

2.1.3 External Inputs (EIs)

External Inputs is that function type that deals with operation that has to elaborate data coming from the external environment. The PowerEnjoy application supports inputs within different interactions with the two types of user that can use the application. All the inputs coming from

users must be processed by the Request manager and since this component does not elaborate the inputs but only invoke the right components in the controller, we omit it in the following description.
All people accessing the platform can interact through:

- Registration: This simple operations, at the level of the business logic components only involve the activity of the Login/ Register Manager. For this reason it can be considered as simple and it only account for 3 FPs.

While the interactions in which the system receives data from an already registered user, are the ones belonging to the following list:

- Login/ Logout: This is a simple operation that interacts with the Registration/ Login Manager and DBMS. For this reason it can be given a value of 6 FPs, considering 3 for each one of the two.

- Book car/ Cancel Booking: those operations are more complex since they require the action of The booking manager, the creation of records in the database and also an interaction with the external existing system. Given their complexity we estimate for this point a total of 12 FPs equally distributed between the two operations.

- Open car: this operation, as it was said for the precedent one is to be considered quite complex and for this reason is labeled with 5 FPs.

- Conclude rent: this operation is the most complex between the ones offered to the users. This requires our system to employ the ride manager, the price and discount manager, to work on the DBMS and to interact with the APIs exposed by the external billing system. For this reason its extimated cost is considered to be 8.

External Inputs	Complexity	FPs
Register	Low	3
Login	Low	3
Logout	Low	3
Book Car	Low	4
Cancel Booking	Low	4
Open Car	Low	4
Conclude Rent	Low	8
	Total	29

Figure 7: External Inputs, Resume Table

2.1.4 External Inquiries (EQs)

The third functional type is External Inquiries and consists of all those operations that involve both the acquisition of an input and the creation of an output. In the case of our application there are mainly three kind of interactions that the registered user can do with the system that can be considered under this label:

- View cars available in one zone: this operation is quite light-weighted from the point of view of the business logic of the application. It involves only the resource manager and the DBMS, and it can be considered a simple inquiry. For this reason it is given 3 FPs.
- View the detailed of a specific car: this can be considered a basic operation. It is served by the resource manager and involves a simple interaction with the database. According to this we give it a value of 3 FPs.
- View Account Details: this operations, as the ones above mentioned, is easy to handle by the system and is composed by a straight forward query on the database performed by the resouce manager. Seen the low complexity of the business-logic that supports it, the operation is considered for a value of 3 FPs.

External Inquiries	Complexity	FPs
View Cars Available	Low	3
View Car's Detail	Low	3
Conclude Rent	Low	3
	Total	9

Figure 8: External Inquiries, Resume Table

2.1.5 External Outputs (EOs)

Under the Functional Type called External Outputs, are grouped those operations in which the system generates data for the external environment. In the case of the PowerEnjoy, this kind of operation are served by the functionality called notify, provided by the request manager. Since this function is then called in many of the routines that are executed on the system we give to it a complexity weight of Average with a corresponding assignment of 7 function points.

External Outputs	Complexity	FPs
Notify	High	7
	Total	7

Figure 9: External Outputs, Resume Table

2.1.6 Overall Estimation

Given the above calculated complexities for each single element weighted according to the tables above mentioned we have now the function points of all the types. They are resumed in the following table: For the method of the function types have been defined tables (derived once more from statistical analysis) to map the number of function points of an application into the estimate number of lines of source code needed to implement the business logic of the considered application, depending on the language used for the development. The figures that we found for the mapping of one function point into lines of JEE code are a lower bound of 46 and an upper

Function Type	FPs
Internal Logic Files	55
External Logic Files	50
External Inputs	29
External Inquiries	9
External Outputs	7
Total	150

Figure 10: Overall Function Points, Resume Table

bound of 67. This means that for each function point will be needed between 46 and 67 lines of code, in order to actually implement it. Thus, we can comput the estimated number of lines of code for the PowerEnjoy application as:

Lower Bound	$\mathbf{SLOC_{lb}}$ = $MappingFactor_{lb}$ x FunctionPoints = 46 * 150 = 6900 Lines of Code
Upper Bound	$\mathbf{SLOC_{ub}}$ = $MappingFactor_{ub}$ x FunctionPoints = 67 * 150 = 10050 Lines of Code

The estimated number of source lines of code (SLOC) varies, for this reason, in the range between 6900 and 10050.

2.2 Cost and Effort Estimation: COCOMO II

As announced in the Introduction in this section we are going to estimate the cost and effort that will be needed in order to successfully design and implement the application for the PowerEnjoy service. In order to do so we are going to follow the COCOMO II approach.

2.2.1 Scale Drivers

Before listing the values for all the scale drivers, as they have been computed by the team, we attach the COCOMO II table for the scale factors, extracted from the official documentation.

Scale Factors	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	The estimated Equivalent Process Maturity Level (EPML) or					
	SW-CMM Level 1 Lower	SW-CMM Level 1 Upper	SW-CMM Level 2	SW-CMM Level 3	SW-CMM Level 4	SW-CMM Level 5
SF_i:	7.80	6.24	4.68	3.12	1.56	0.00

Here it follows a brief description for each of the scale drivers that have been analyzed:

- **Precedentedness:** Since the components of our team have worked on other similar projects together although we can't consider ourselves experts in this kind of project, we will give a **nominal** precedentedness value.
- **Development Flexibility:** Our project has been scheduled and guided by a set of deadlines and requirements provided by software engineering 2 course, but its also fair to state that the implementation and description of the system structure has been build by us nearly from scratch, meaning that we've been able to make our assumptions and choose the technologies to use. For this reason this section will have a value of **low**.
- **Risk Resolution:** The percent of development schedule devoted to establish an architecture is high, moreover, the risk analysis we performed pretends to treat nearly all sections of our work. Our team of qualified software architects has also been using specific support tools to reduce the amount of uncertainty in key architecture drivers. Hence this value will be set to **very high**.
- **Team Cohesion:** Our team members know each other very well and have experience working together. This coefficient is set to **extra high**.
- **Process Maturity:** Although our project accomplishes all required goals, we have seen that there's a part of our system concerning the database distribution architecture in internal and external systems that should be reviewed to decide which option is optimal. For this reason we will give this section a **level 4**.

Scale Driver	Factor	Value
Predentedness (PREC)	Nominal	3,72
Development flexibility (FLEX)	Low	4,05
Risk Resolution (RESL)	Very High	1,41
Team Cohesion (TEAM)	Extra High	0,00
Process Maturity (PMAT)	Level 4	1,56
	Total	10,74

2.2.2 Cost Drivers (post-Architecture)

- **Required Software Reliability:** Our system has the bank details of users and manages fees and billings, for this reason it requires a **high** reliability.

RELY Descriptors:	slight inconven-ience	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

- **Database Size:**Database size: Our estimations reach the 3 GB of data for our internal database. Hence the DATA coefficient is **very high** given a value of 8700 as SLOC.

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

- **Product Complexity:**
  - **Control Operations:** The operations carried out by the whole system could reach a high complexity level, but since we made the assumption that most of our operations are handled by the existing system we will set this section to a **nominal** value.
  - **Computational Operations: nominal**
  - **Device Dependent Operations: nominal**
  - **Data Management Operations:** Our application will use simple triggers in its database. And since the database of the external and the internal system is not the same, they will also need some kind of synchronization. This value is set to **very high**.
  - **User Interface Management Operations:** This value is set to **high** because the application will use extensions such as google maps to represent the cars apart from possibly developing a widget set for the DBMS.

  Computing the average of the previous coefficients, we will give a **nominal** value to Product complexity.
- **Required Reusability:** The required reusability is **high** because we are extending and reusing parts of an already made product, and it is fairly reasonable to expect the usage of the application in a future.

RUSE Descriptors:		none	across project	across program	across product line	across multiple product lines
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	0.95	1.00	1.07	1.15	1.24

- **Documentation Match to life Cycle Needs:** From our point of view, it is right sized to life-cycle needs, thus we assign a rate of **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

- **Execution Time Constraint:** We will set this value to **nominal** because although the software is complex, the time dedicated to its operations is relatively small.

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

- **Main Storage Constraint:** Our system doesn't require significant amount of storage, this value is set to **nominal**.

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

- **Platform Volatility:** Keeping in mind that we are developing a software including a DBMS, that depends on two other external systems, this value should be between nominal and high, so we will set it to **high**.

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

- **Analyst Capability:** The requirements and design features have been built following high-level procedures and references by the team.

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

- **Programmer Capability:** The team programming skills should be considered as **high**.

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

- **Personnel Continuity:** The value is set to **very low** because all our team will be replaced.

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	

- **Applications Experience:** The value is set to **low** because the team has more than 6 months of experience on the majority of the project sections, but not all of them.

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

- **Platform Experience:** The experience of our team with the used platforms is to be considered **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

- **Language and Tool Experience:** This value is set to **nominal** for our team.

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	

- **Use of Software Tools:** The extension and variety of the system demands a **high level** in this section.

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

- **Multisite Development:** The site collocation should be the same city, at least for the part of system that we are developing, and the communication support correspond to a **high** level as well.

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

- **Required Development Schedule:** Although we have spent a significant amount of time planning and designing the project, we also expect to dedicate a large amount of time to development, for this reason we give it a value of **nominal**.

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

And finally we attach the resume table of the cost driver evaluation:

Scale Driver	Factor	Value
Required Software Reliability (RELY)	High	1,1
Database Size (DATA)	Very High	1,28
Product Complexity (CPLX)	Nominal	1,0
Required Reusability (RUSE)	High	1,07
Documentation Match to Life Cycle Needs (DOCU)	Nominal	1,0
Execution Time Constraint (TIME)	Nominal	1,0
Main Storage Constraint (STOR)	Nominal	1,0
Platform Volatility (PVOL)	High	1,15
Analyst Capability (ACAP)	High	0,85
Programmer Capability (PCAP)	High	0,88
Personnel Continuity (PCON)	Very Low	1,29
Application Experience (APEX)	Low	1,1
Platform Experience (PLEX)	Nominal	1,0
Language and Tool Experience (LTEX)	Nominal	1,0
Use of Software Tools (TOOL)	High Level	0,9
Multisite Development (SITE)	High	0,93
Required Development Schedule (SCED)	Nominal	1,0
	Total	1,408

2.2.3 Effort Equation

After the above reported evaluations is possible to use the effort estimation in Person-Months (PM):

$$\text{Effort} = \text{A} * \text{EAF} * \text{KSLOC}^{\text{E}}$$

where:

$$\text{A} = 2.94 \text{ (for COCOMO II)}$$

$$\text{EAF} = [\text{product of all cost drivers}] = 1.5391$$

$$\text{E} = [\text{exponent derived from the scale drivers}] =$$
$$= \text{B} + 0.01 * \sum_i SF_i = \text{B} + 0.01 * 10.74 = 1.0174$$

in which B is equal to 0.91 for COCOMO II

Thanks to the above computed parameters we can compute the effort estimation respectively for the lower and upper bound given for the SLOCs:

$$\text{Effort}_{lb} = A * EAF * {KSLOC_{lb}}^{\text{E}} = 2.94 * 1.5391 * 6.9^{1.0174} = 32.29PM$$

$$\text{Effort}_{ub} = A * EAF * {KSLOC_{ub}}^{\text{E}} = 2.94 * 1.5391 * 10.05^{1.0174} = 47.34PM$$

And a consequent Schedule estimation given by the following formula (again both for the lower and the upper bound):

$$\text{Duration} = 3.67 * \text{Effort}^{\text{F}}$$

where:

$$\text{F} = 0.28 + 0.2 * (\text{E-B}) = 0.28 + 0.2 * 0.1074 = 0.3014$$

And from that we can compute the upper and lower bound for the estimated duration of the project:

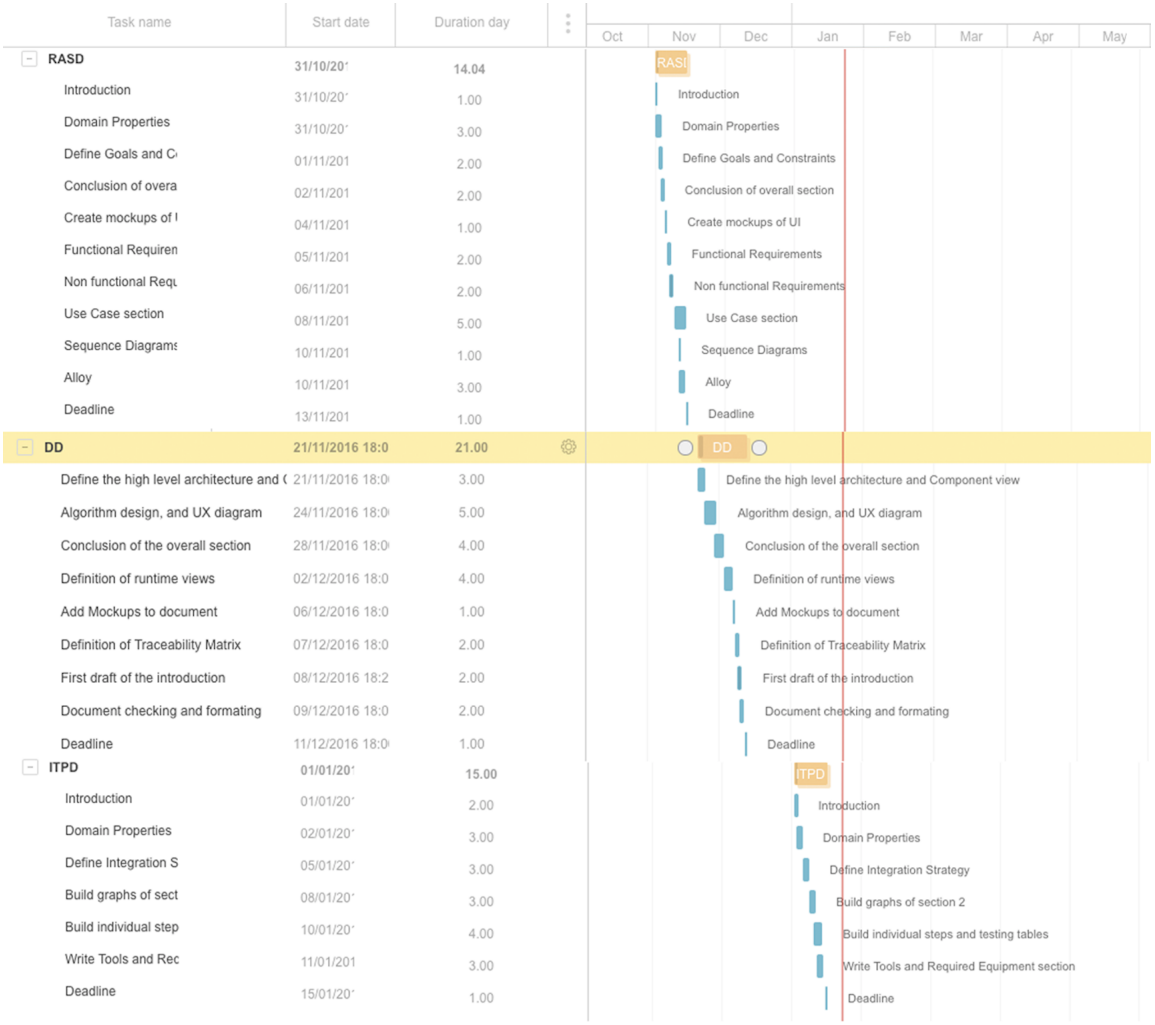
$$\text{Duration}_{lb} = 3.67 * {Effort_{lb}}^{\text{F}} = 3.67 * 32.29^{0.3014} = 10.45Months$$

$$\text{Duration}_{ub} = 3.67 * {Effort_{ub}}^{\text{F}} = 3.67 * 47.34^{0.3014} = 11.73Months$$

In conclusion we can assume that the effort necessary to carry out the project is included in the interval between 32.29 and 47.34 person per month and thus, the estimated duration of the project varies between 10.45 and 11.74 months.

3 Schedule

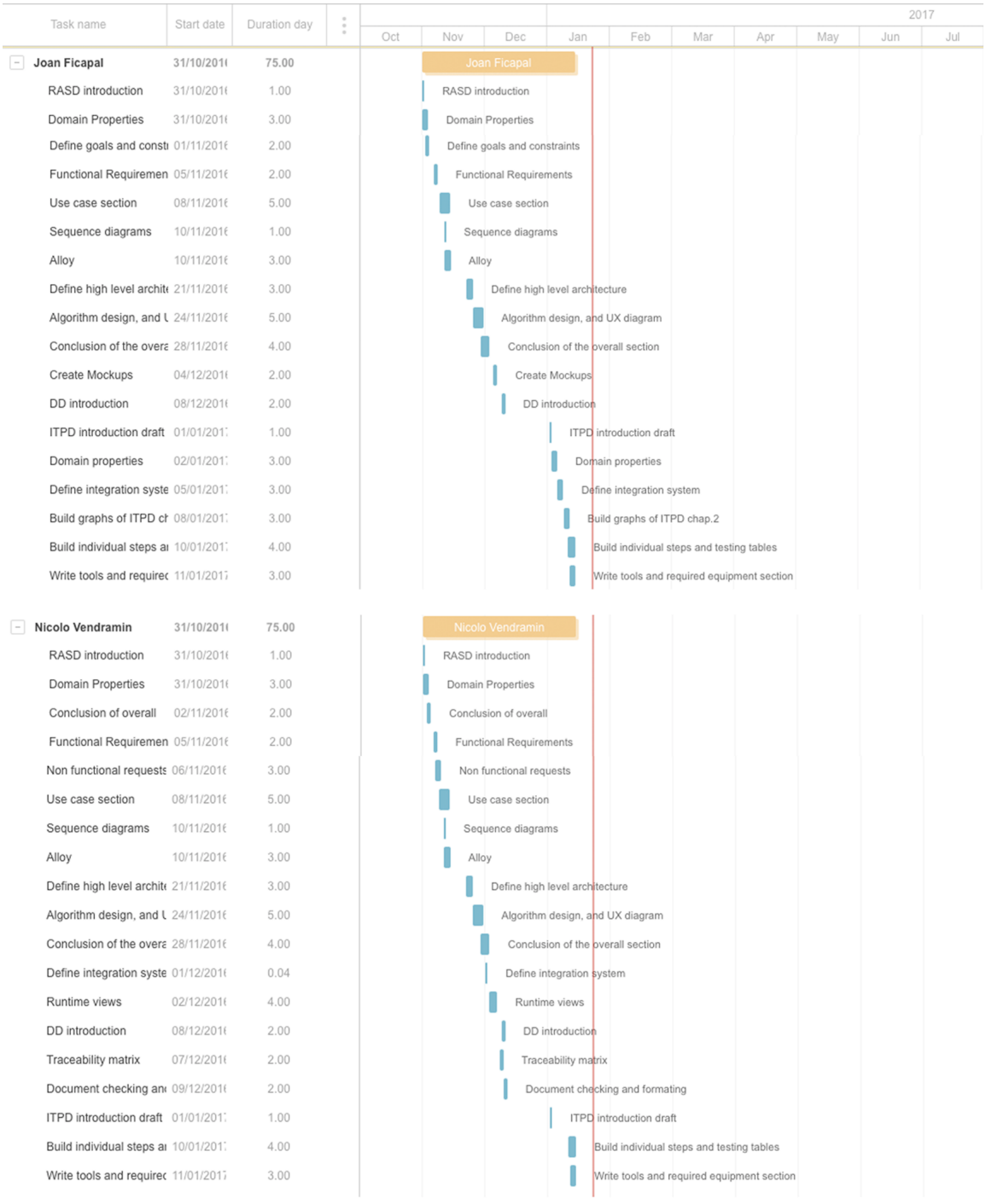
In the following section we are going to provide an high-level project schedule.
Although the project is made for didactic purpose and no development is going to take place, we are going to include in the following schedule the steps that we will not actually do.
Since the dimension of the schedule was quite big it has been splitted in two parts: the first represents the activities actually performed by the team. The second instead is the part of the schedule concerning the future development of the PowerEnjoy application.




Task name
Start d...
Duration day
2017
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Today
Total estimate 19/01/20 203.13
Development 19/01/20 203.13
Subcomponent deve 19/01/20 162.33
Code inspection 01/02/20 120.08
Unit tests 01/04/20 15.00
Subcomponent integ 13/04/20 18.00
Component develop 01/05/20 20.00
Component integrat 21/05/20 15.00
Integration testing 05/06/20 7.00
System testing 12/06/20 5.00
Refinements 12/06/20 14.00
App development 26/06/20 1.00
Demonstration to st 27/06/20 1.00
Refinements 28/06/20 15.00
Internal demonstrati 11/07/201 1.00
Second demonstrati 12/07/20 1.00
Refinements 13/07/20 15.00
Deep revision docur 17/07/20 7.00
Final presentation to 24/07/20 1.00
Final refinements 24/07/20 15.00
User manual 31/07/20 10.00
Release 09/08/20 1.00
Deployement 09/08/20 23.00
System Deployme 09/08/20 5.00
Application distribu 10/08/20 3.00
Final meetings 10/08/20 7.00
Final preparations 28/08/20 4.00
Start - up 28/08/20 33.00
System first start 28/08/20 10.00
Real life simulatior 30/08/20 15.00
Main updates and 31/08/20 30.00
Development
Subcomponent development
Code inspection
Unit tests
Subcomponent integration
Component development
Component integration
Integration testing
System testing
Refinements
App development
Demonstration to stakeholders
Refinements
Internal demonstration
Second demonstration to stakeholders
Refinements
Deep revision document
Final presentation to stakeholders
Final refinements
User manual
Release
Deployement
System Deployment
Application distribution
Final meetings
Final preparations
Start - up
System first start
Real life simulation
Main updates and checks

4 Resource Allocation

In this chapter we are going to attach an outline of how the tasks defined in the schedule that is given in the previous section have been splitted between the two members of the development team. This document must be considered only as a general overview, and more detailed schedules will be produced during the the project to specify in major detail the organization and split of the work in each of the development phases. The diagram is split in two parts because of its size and because from the development phase on, the development of the project will be assigned to an external development team, which differs from the team that is working on the project at the moment.


Task name
Start date
Duration day
2017
Oct
Nov
Dec
Jan
Feb
Mar
Apr
May
Jun
Jul
Joan Ficapal 31/10/2016 75.00
RASD introduction 31/10/2016 1.00
Domain Properties 31/10/2016 3.00
Define goals and consti 01/11/2016 2.00
Functional Requiremen 05/11/2016 2.00
Use case section 08/11/2016 5.00
Sequence diagrams 10/11/2016 1.00
Alloy 10/11/2016 3.00
Define high level archite 21/11/2016 3.00
Algorithm design, and U 24/11/2016 5.00
Conclusion of the overa 28/11/2016 4.00
Create Mockups 04/12/2016 2.00
DD introduction 08/12/2016 2.00
ITPD introduction draft 01/01/2017 1.00
Domain properties 02/01/2017 3.00
Define integration syste 05/01/2017 3.00
Build graphs of ITPD ch 08/01/2017 3.00
Build individual steps a 10/01/2017 4.00
Write tools and require 11/01/2017 3.00
Joan Ficapal
RASD introduction
Domain Properties
Define goals and constraints
Functional Requirements
Use case section
Sequence diagrams
Alloy
Define high level architecture
Algorithm design, and UX diagram
Conclusion of the overall section
Create Mockups
DD introduction
ITPD introduction draft
Domain properties
Define integration system
Build graphs of ITPD chap.2
Build individual steps and testing tables
Write tools and required equipment section
Nicolo Vendramin 31/10/2016 75.00
RASD introduction 31/10/2016 1.00
Domain Properties 31/10/2016 3.00
Conclusion of overall 02/11/2016 2.00
Functional Requiremen 05/11/2016 2.00
Non functional requests 06/11/2016 3.00
Use case section 08/11/2016 5.00
Sequence diagrams 10/11/2016 1.00
Alloy 10/11/2016 3.00
Define high level archite 21/11/2016 3.00
Algorithm design, and U 24/11/2016 5.00
Conclusion of the overa 28/11/2016 4.00
Define integration syste 01/12/2016 0.04
Runtime views 02/12/2016 4.00
DD introduction 08/12/2016 2.00
Traceability matrix 07/12/2016 2.00
Document checking an 09/12/2016 2.00
ITPD introduction draft 01/01/2017 1.00
Build individual steps a 10/01/2017 4.00
Write tools and require 11/01/2017 3.00
Nicolo Vendramin
RASD introduction
Domain Properties
Conclusion of overall
Functional Requirements
Non functional requests
Use case section
Sequence diagrams
Alloy
Define high level architecture
Algorithm design, and UX diagram
Conclusion of the overall section
Define integration system
Runtime views
DD introduction
Traceability matrix
Document checking and formating
ITPD introduction draft
Build individual steps and testing tables
Write tools and required equipment section


Task name
Start d...
Duration day
2017
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Today
Total estimate 19/01/20 203.13
External Developmen 19/01/20 203.13
Subcomponent deve 19/01/20 162.33
Code inspection 01/02/20 120.08
Unit tests 01/04/20 15.00
Subcomponent inte 13/04/20 18.00
Component develop 01/05/20 20.00
Component integrat 21/05/20 15.00
Integration testing 05/06/20 7.00
System testing 12/06/20 5.00
Refinements 12/06/20 14.00
App development 26/06/20 1.00
Demonstration to st 27/06/20 1.00
Refinements 28/06/20 15.00
Internal demonstrati 11/07/201 1.00
Second demonstrati 12/07/20 1.00
Refinements 13/07/20 15.00
Deep revision docur 17/07/20 7.00
Final presentation tc 24/07/20 1.00
Final refinements 24/07/20 15.00
User manual 31/07/20 10.00
Release 09/08/20 1.00
Deployement Team 09/08/20 23.00
System Deployme 09/08/20 5.00
Application distribu 10/08/20 3.00
Final meetings 10/08/20 7.00
Final preparations 28/08/20 4.00
Start - up Team 28/08/20 33.00
System first start 28/08/20 10.00
Real life simulatior 30/08/20 15.00
Main updates and 31/08/20 30.00
External Development Team
Subcomponent development
Code inspection
Unit tests
Subcomponent integration
Component development
Component integration
Integration testing
System testing
Refinements
App development
Demonstration to stakeholders
Refinements
Internal demonstration
Second demonstration to stakeholders
Refinements
Deep revision document
Final presentation to stakeholders
Final refinements
User manual
Release
Deployment Team
System Deployment
Application distribution
Final meetings
Final preparations
Start - up Team
System first start
Real life simulation
Main updates and checks

5 Risk Management

In these section we are going to outline which are the main risks that the development of the PowerEnjoy project may face.
Generally speaking the problems that we could face can be grouped in x groups:

- Problems with the staff
- Problems with the stakeholders
- Technical Problems
- Managerial Problems
- Non Forecastable events affecting the evolution of the project.

Problems belonging to the first category are mainly due to the strong turnover that the personnel working on the project could suffer. This turnover is caused by the fact that the development of the project will be assigned to a group of developer selected on purpose that may need some time to integrate in the project, causing in that way delays on the planned schedule, and also by the fact that the job-market in the field of IT is a quite dynamic market and exists the concrete possibility that some members of the staff leave the company. With managerial problems we refer to the ineptitude of the managers leading the team in the different phases of development (from the elicitiation to the deployment and maintenance of the project). A bad attitude of the team leader could affect all the team, causing delays, an inefficient use of resources and possibly a bad working environment.
The second category is wide in the sense that the stakeholders of the project are many and the possible problems that it could face because of them. We are going to list the main ones, along with a quick explenation:

- Customer not interested in the project: even if some studies about the attractiveness of the applications have been done before the project was commissioned, a small, but not insignificant, probability that the final customer won't like the service is still to be considered.
- Problems with direct competitors: we know that in the same market, some companies are already operating with profit. Even if in this kind of market a new service can be seen by the client as an appreccciated integration of the coverage provided by the other services, is as well true that the presence of incumbents well-established in the market could cause a scarse interest in the client. In addition the incumbents could apply some moves to make sure that the project fails without affecting their business.
- Problems with indirect competitors: with indirect competitors (e.g.: Public Transportation, bike-sharing services, the taxi service...) it could happen that in order to preserve their business they try to make pressure on the local governance in order to reduce the gap between them and the car-sharing service, affecting in this way the possible success of the PowerEnjoy project.
- Problems in the relationship with the local governance: in order to run the service, permissions and sponsorship must be granted by the local governance of the city. In case the relationship goes wrong for any possible reason the project could suffer for delays in the schedule, due to the non-collaboration of the local authorities.

In order to avoid problems with the stakeholders in general is suggested to try to include them in the process of development of the project so that the relationship with them will be kept in the best possible way.
The third possible category is the one concerning technical problems. The main possible problem is connected to the functioning and battery-duration of the cars used to provide the service. In case the vehicles chosen to be employed in the service are not sufficiently good the service would be perceived as non usefull, this being a problem for it. And finally the last category is mentioned to remind that the schedule could be affected as well by unforecastable events.