PowerEnJoy, Project Plan

Riccardo Cattaneo 873647
Fabio Chiusano 874294

Version: 1.0.0
Release date: 03-01-2017

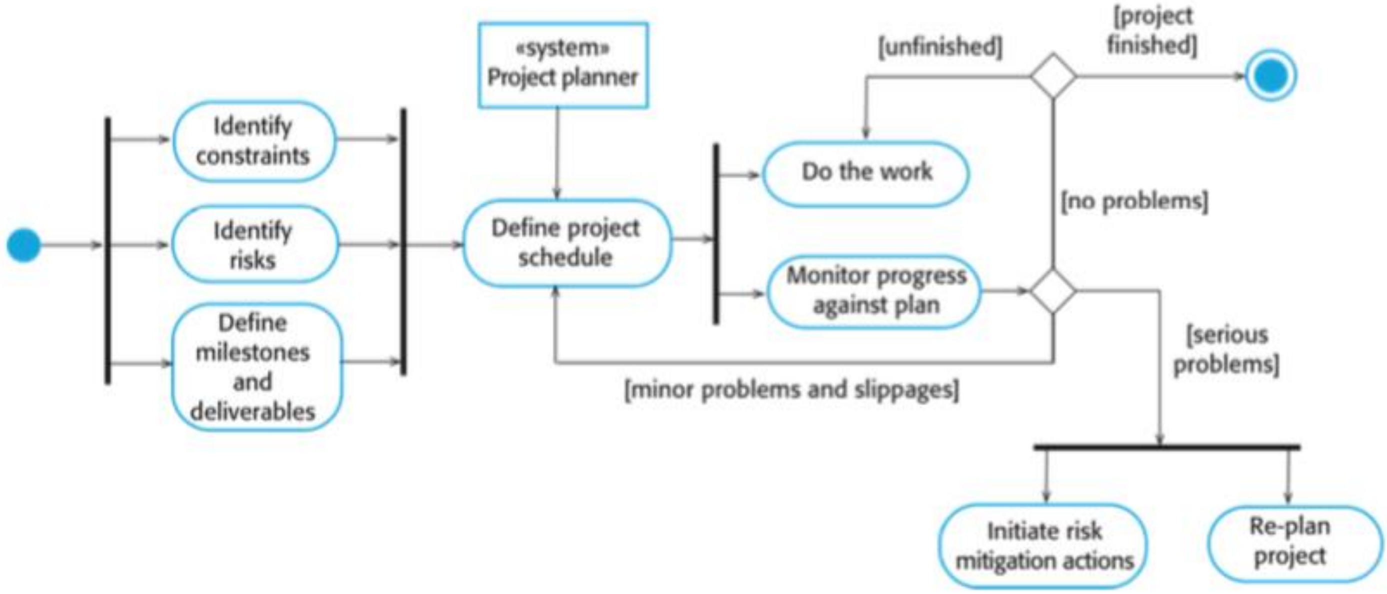
Table of Contents

1 Introduction

1.1 Purpose and Scope

Project planning is an iterative process that starts in the project startup phase. Plan changes are inevitable as more information about the system to build become available during the development and business goals change.

This is an overview of the project planning process:



This Project Plan Document aims at:

- Estimating the overall project size;
- Thus, estimating the effort needed and the time required to project, develop and test the PowerEnjoy System;
- Identifying all the possible risks, and the related actions that should be taken;
- Defining project schedule and the division of tasks among team members.

1.2 List of Definitions and Abbreviations

- BL: Business Logic;
- ITPD: Integration Test Plan Document;
- DD: Design document;
- RASD: Requirements Analysis and Specification Document;
- COCOMO II: COnstructive COst MOdel, version II.2000.0;
- KSLOC: Kilo Source Line Of Code;
- FP: Function Point;

- UFP: Unadjusted Functional Point;
- ILF: Internal Logic File;
- ELF: External Logic File;
- PM: Person Month;
- Project task: activity that must be completed in order to fulfill the project;

1.3 List of Reference Documents

- Specification Document: Assignments AA 2016-2017.pdf;
- RASD Document v1.1;
- Design Document v1.1;
- ITPD Document v1.0;
- Example of Project Reporting Documents from previous years;

2 Size estimation: Function Points

2.1 Introduction

We use the Functional Point approach in order to assess the final dimension of the project. It is based on the evaluation of the functionalities of the software to be, that have been taken from the RASD Document. It's possible to convert Function Points to KSLOC (i.e. thousands of lines of source code), which will be used in COCOMO to evaluate the effort needed to accomplish the system.

Function Points evaluations depend on the estimator and therefore they are very subjective. Therefore, automatic FP counting is impossible and the results must be accepted with their uncertainties.

The FP types are:

- Internal Logic File: it represents a set of homogeneous data handled by the system;
- External Interface File: it represents a set of homogeneous data used by the application but handled by external application;
- External Input: elementary operation that allows input of data in the system;
- External Output: elementary operation that creates a bitstream towards the outside of the application;
- External Inquiry: elementary operation that involves input and output operations.

The following table outlines the types of Functional Points along with their weight, according to their complexity. The total number of FPs can be computed as the weighted sum of function types using the coefficient listed below:

Function Type	*Complexity*		
	Simple	*Medium*	*Complex*
Internal Logic File	7	10	15
External Logic File	5	7	10
External Input	3	4	6
External Output	4	5	7
External Inquiry	3	4	6

2.2 Function Points Estimation

2.2.1 Internal Logic Files

The system includes a number of ILFs that will be used to store the information about:

- Users: few simple information, such username, password, email;
- Cars: static information such as model of the car and seats, dynamic information such as battery level and availability. Because of the communication and the synchronization with the remote system of the car, we have considered this point as high complexity;
- Reservations: the system has to keep updated information about each reservation, with related start time and start area, car and user involved, and eventually final cost, obtained with the help of data from the car;
- Rides: information about starting and releasing time of a ride and release area;
- Safe Areas: information on location of a safe area, possibly the possibility of charging a car.

ILF	*Complexity*	*FPs*
User	Low	7
Car	High	15
Reservation	Medium	10
Ride	Medium	10
Safe Area	Low	7
	Total	49

2.2.2 External Logic Files

The system has to interact with two external handlers:

- Payment Handler: the interaction is simple, because the more complex operations are all performed by the external handler, the system has only to receive and store information;
- Search-on-a-map Handler: interaction is more frequent and complex, the system possibly overlays information on received map;

ELF	*Complexity*	*FPs*
Payment	Medium	7
Map	Medium	7
	Total	14

2.2.3 External Inputs

The system interacts with the user to allow him/her to:

- Login/logout: these are simple operations, so we can adopt the simple weight for them;
- Sign-up: validation of payment is performed of external handler, we can adopt the simple weight for this operation;
- Change Payment Info: as above, we have considered it as a simple operation;
- Search for an address: the user inserts an address manually and the system shows a map centred on that address. Both the manual insertion and the GPS position are managed maintaining simple interaction between components;
- Search of near available Cars: the user can search for Safe Areas with almost one available car that are near a certain address. The process of this request involves fairly complex algorithm in order to optimize the search performance;
- Search of near safe areas: the user can search for Safe Areas and Special Safe Areas;
- Reserve an available car: The user selects one car to be reserved among the available ones. This operation implies multiple interaction: for example, the car has to be marked as unavailable, a reservation code has to be shown in the car dashboard and the expiration of the reservation has to be checked;
- Unlock a car: the user inserts the code shown on the car dashboard in order to unlock the car he/she has just reserved, we have considered it as medium complexity because it implies synchronization between multiple components.

Ext. Input	*Complexity*	*FPs*
Login/logout	Low	2*3
Sign-up	Low	3
Change Payment Info	Low	3
Search for an address	Medium	4
Search near cars	High	6
Search near Safe Areas	Medium	4
Reserve an available car	High	6
Unlock a car	Medium	4
	Total	36

2.2.4 External Outputs

- Ride Conclusion: after that the user has concluded a ride, the system updates ride and reservation entities and calculates the final discharged cost that the user has automatically payed. These data are saved and stored in order to allow the user to consult order history;
- Reservation expired: if the user doesn't pick up the reserved car in time, the system should mark the reservation as expired, apply the expiration fee and make the reserved car available again.

Ext. Output	*Complexity*	*FPs*
Ride Conclusion	Medium	5
Reservation expired	Medium	5
	Total	10

2.2.5 External Inquiries

- Reservation History: each user can consult his/her personal reservation history. This process does not imply complex algorithms;
- Retrieve Safe Area info: after the user has selected a safe area, the system shows information about available cars in that area and possibly power grids status. Since it implies interaction of multiple entities, we have considered it as medium complexity;
- Show personal info: each user can consult his/her personal info (i.e. his/her information stored in the PowerEnjoy system).

Ext. Inquiry	*Complexity*	*FPs*
Reservation History	Low	3
Retrieve Safe Area info	Low	3
Show Personal info	Low	3
	Total	9

2.3 Total FP number and summary

Function Type	*Unadjusted Functional Points*
Internal Logic File	49
External Logic File	14
External Input	36
External Output	10
External Inquiry	9
Total	118

By summing up all these numerical values we get a total estimation of 118 UFP.

It is possible to estimate a number of line of code from the number of functional points. We have used the value present in the table found at this URL: http://www.qsm.com/resources/function-point-languages-table

[SLOC/UFP] J2EE: 46

The total estimated number of KSLOC is:
KSLOC/UFP * UFP = 0.046 * 118 = **5.428 KSLOC**

3 Effort and cost estimation: COCOMO II

3.1 Introduction

Thanks to COCOMO, we can estimate the amount of effort needed to accomplish the finished system.

This estimation, whose unit of measure in the Person Month, is achieved through a complex, non-linear model that takes in account the characteristics of the product but also of people and process.

The general equation that calculates the effort needed is:

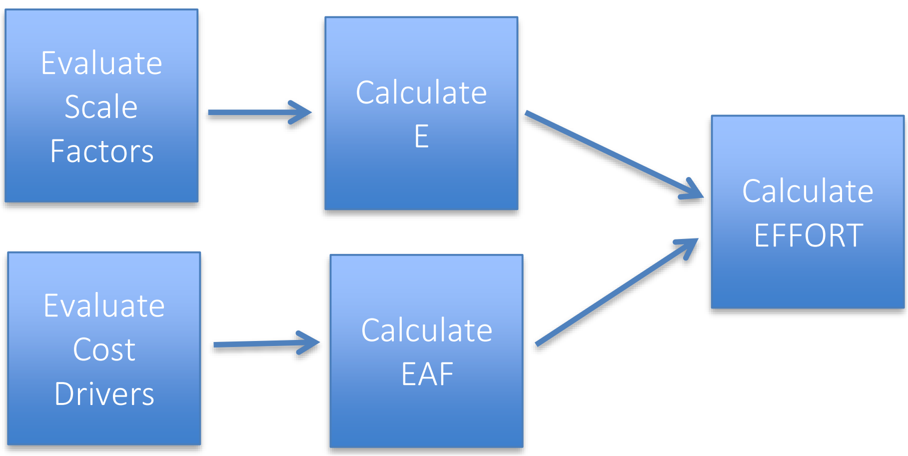
$$EFFORT = A * SIZE^{E} * EAF,$$

where:

- **EFFORT** stands for the estimated number of Person Month for the project, that is the estimated project effort;
- **A** is an organisation-dependent constant. We'll use the value **2.94**, as suggested by COCOMO II.2000;
- **SIZE** is the project KSLOC. We'll use the size estimated previously with FP, which is **5.428 KSLOC**;
- **E** reflects the disproportionate effort for large project and is derived from Scale Factors;
- **EAF** (Effort Adjustment Factor) is the product of all the Cost Drivers.

Scale Factors and Cost Drivers are defined in the following sections.

The following diagram summarizes the steps for estimating the effort:



All the tables used in this analysis have been taken from COCOMO II, Model Definition Manual at:
http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf

3.2 Scale Factors

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_j:	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_j:	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_j:	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_j:	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_j:	7.80	6.24	4.68	3.12	1.56	0.00

These are the five Scale Factors with our evaluation for this project:

- Precedentedness: It reflects the previous experience that we have with this kind of projects. We have different past experience, but generally this is the first experience using this framework and these development methodologies, this value will be Low;
- Development Flexibility: It reflects the degree of flexibility in the development process. The client has only set some clear general goals, so it is possible to classify the factor as High;
- Risk Resolution: Reflects the extent of risk analysis carried out. Thanks to Risk and Recovery Actions analysis carried out in the following chapters of this document, this factor can be considered as High;
- Team Cohesion: Team members know each other very well, thus the cooperation can be proactive. All the stakeholders seem to have the same vision, so we have evaluated this factor as High;
- Process Maturity: Reflects the process maturity of the organisation. We have concluded that this factor can be evaluated as Nominal.

Results are summarized in the following table:

Scale Factor	*Evaluation*	*Value*
Precedentedness	Low	4.96
Development flexibility	High	2.03
Risk resolution	High	2.83
Team cohesion	High	2.19
Process maturity	Nominal	4.68
	Total	16.69

It is now possible to calculate the parameter E of the Effort Formula, using the following:

$$\mathbf{E = B + 0.01 \times \sum_{1<=j<=5} SF_j}\ ,$$

where:

- B is a constant whose value is 0.91, as suggested by COCOMO II.2000;
- SF_j is the value of the j-th Scale Factor.

Hence E = 1.0769.

3.3 Cost Drivers

We have to consider Cost Drivers for an Early Design System, because the PowerEnjoy system has to be built from the ground up.
Early design evaluation implies the combination of multiple cost drivers arising from Post Architecture Estimation, following this table:

Early Design Cost Driver	Counterpart Combined Post-Architecture Cost Drivers
PERS	ACAP, PCAP, PCON
RCPX	RELY, DATA, CPLX, DOCU
RUSE	RUSE
PDIF	TIME, STOR, PVOL
PREX	APEX, PLEX, LTEX
FCIL	TOOL, SITE
SCED	SCED

We will now calculate the value of each early design cost driver by evaluating the corresponding post-architecture cost drivers. Each of these has a rating scale from Very Low (=1) to Very High (=5). Adding up their numerical ratings produces a value, that corresponds to a certain Rating Level with related Effort Multiplier.

3.3.1 Personnel Capability (PERS)

- Analyst Capability (ACAP) – High:

Table 26. ACAP Cost Driver

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

Design and analysis abilities should be set to high, since a lot of effort will be dedicated in analysing the problem specification and requirements, drafting the RASD Document, designing the hardware and software architecture and drafting the Design Document. Moreover, most of the ambiguities present in the initial description will be detailed and resolved in RASD, document that all the stockholders have to approve.

- Programmer Capability (PCAP) – High:

Table 27. PCAP Cost Driver

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

This parameter is evaluated according to our degree of cooperation, which is run in. Thus, the value is set to high.

- Personnel Continuity (PCON) – Low:

Table 28. PCON Cost Driver

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	

This parameter is relevant in particular since in the current case our available time is less than half a year and we can't dedicate all our time to the project. For this reason, we have set it to low.

Table 36. PERS Cost Driver

PERS Descriptors:							
• Annual Personnel Turnover	45%	30%	20%	12%	9%	6%	4%
Rating Levels	Extra Low	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	2.12	1.62	1.26	1.00	0.83	0.63	0.50

The total numeric value is 4+4+2=10.
This corresponds to a High Rating Level, and its related effort multiplier is 0.83.

3.3.2 Product Reliability and Complexity (RCPX)

- Required software reliability (RELY) – Nominal:

Table 17. RELY Cost Driver

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

A software failure would not cause risks to human life, but would probably cause some financial problems.

- Database size (DATA) – High:

Table 18. DATA Cost Driver

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

The size of data contained in the database will be probably quite big compared to the number of line of code. Thus, we have set this parameter as High.

- Product complexity (CPLX) – Nominal:

Table 20. CPLX Cost Driver

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

We have evaluated the complexity of control operations, computational operations, device-dependent operations, data management operations and user interface management operations, using the table provided in the COCOMO definition manual. We have evaluated the overall complexity as Nominal.

- Documentation match to life-cycle needs (DOCU) – High:

Table 22. DOCU Cost Driver

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

The completeness of documents that will be produced is high, but also the time and efforts required for drawing up them. Although this choice will reduce time and effort needed in other part of the project such as maintenance, because of the better understanding of project and software that all the documents guarantee.

Table 37. RCPX Cost Driver

RCPX Descriptors:							
Rating Levels	Extra Low	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.49	0.60	0.83	1.00	1.33	1.91	2.72

The total numeric value is 3+4+3+4=14.
This corresponds to a High RCPX Rating Level, and its related effort multiplier is **1.33**.

3.3.3 Developed for Reusability (RUSE)

Table 21. RUSE Cost Driver

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

This Early Design model cost driver is the same as its Post-Architecture counterpart. Thanks to the design that follows common software engineering principles, that enhance the reusability of code, we have set this parameter as High. This value fits also the constraints imposed by the value of Reliability and Database Size.

The High Rating Level corresponds to an effort multiplier of **1.07**.

3.3.4 Platform Difficulty (PDIF)

- Execution time constraint (TIME) – Nominal:

Table 23. TIME Cost Driver

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

We have evaluated the execution time expected to be used by the system, consuming the total execution time resource, as Nominal. This because the the remarkable increase in available processor execution efficiency of modern CPUs.

- Main storage constraint (STOR) – Nominal:

Table 24. STOR Cost Driver

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

As for the Time constraint, also the storage constraint is subject to the same considerations. Our system has not any particular storage-angry feature, so it is quite simple to get hardware that will not give concerns about space occupation.

- Platform volatility (PVOL) – Nominal:

Table 25. PVOL Cost Driver

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

Changes in Mobile Systems are frequent, anyway a certain degree of backward compatibility is given. Thus, we have set the platform volatility as Nominal.

Table 38. PDIF Cost Driver

PDIF Descriptors:					
• Sum of TIME, STOR, and PVOL ratings	8	9	10 - 12	13 - 15	16, 17
• Time and storage constraint	≤ 50%	≤ 50%	65%	80%	90%
• Platform volatility	Very stable	Stable	Somewhat volatile	Volatile	Highly volatile
Rating Levels	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.87	1.00	1.29	1.81	2.61

The total numeric value is 3+3+3=9.
This corresponds to a Nominal PDIF Rating Level, and its related effort multiplier is **1.00**.

3.3.5 Personnel Experience (PREX)

- Application experience (APEX) – Low:

Table 29. APEX Cost Driver

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

Our project experience is evaluated according to our previous experience in web projects and also according to our abilities in programming in Java and most importantly in the Java EE framework.
We have different past experiences, but, since we are still students, we can evaluate our experience as a Low value.

- Language and tool experience (LTEX) – Nominal:

Table 31. LTEX Cost Driver

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	

We have started using tools tools involved in software engeneering, such as ones that perform requirements and design representation and analysis, about one years ago. The best level that represents our language and tool experience is Nominal.

- Platform experience (PLEX) - Nominal:

Table 30. PLEX Cost Driver

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

Also our average knowledges about platforms, such as databases, user interfaces and server side development, can be judged as Nominal.

Table 39. PREX Cost Driver

PREX Descriptors:							
Rating Levels	Extra Low	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.59	1.33	1.22	1.00	0.87	0.74	0.62

The total numeric value is 2+3+3=8.
This corresponds to a Low PREX Rating Level, and its related effort multiplier is **1.22**.

3.3.6 Facilities (FCIL)

- Use of software tools (TOOL) – Nominal:

Table 32. TOOL Cost Driver

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

Tools that we will use during the project are, for example, Git, NetBeans, Maven and SonarQube, and others that will be mentioned in the other Documents.
We think that the best level for this cost driver is Nominal, because tools mentioned above are simple basic lifecycle tools that are moderately integrated.

- Multisite development (SITE) – High:

Table 33. SITE Cost Driver

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 communication.	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

This parameter reflects how we handled the distribution of development over distance and multiple platforms.
We think that the best value for our situation is High.

Table 40. FCIL Cost Driver

FCIL Descriptors:							
• Sum of TOOL and SITE ratings	2	3	4, 5	6	7, 8	9, 10	11
• TOOL support	Minimal	Some	Simple CASE tool collection	Basic life-cycle tools	Good; moderately integrated	Strong; moderately integrated	Strong; well integrated
• Multisite conditions	Weak support of complex multisite development	Some support of complex M/S devel.	Some support of moderately complex M/S devel.	Basic support of moderately complex M/S devel.	Strong support of moderately complex M/S devel.	Strong support of simple M/S devel.	Very strong support of collocated or simple M/S devel.
Rating Levels	Extra Low	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.43	1.30	1.10	1.0	0.87	0.73	0.62

The total numeric value is 3+4=7.
This corresponds to a High FCIL Rating Level, and its related effort multiplier is **0.87**.

3.3.7 Required Development Schedule (SCED)

Table 34. SCED Cost Driver

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

Also this Early Design model cost driver is the same as its Post-Architecture counterpart. We plan to distribute our efforts well over the available development time. We have selected the Nominal value for this cost driver.
The Nominal Rating Level corresponds to an effort multiplier of **1.00**.

3.3.8 Summary

Scale Driver	*Factor*	*Value*
Personnel Capability	High	0.83
Product Reliability and Complexity	High	1.33
Developed for Reusability	High	1.07
Platform Difficulty	Nominal	1.00
Personnel Experience	Low	1.22
Facilities	High	0.87
Required Development Schedule	Nominal	1.00
	Product	1.25

3.4 Effort Equation

This final equation gives us the effort estimation measured in Person-Months (PM):

EFFORT = A * SIZE ^{E} * EAF,

where:

- **A = 2.94**, as justified in the introduction of the COCOMO chapter;
- **SIZE = 5.428 KSLOC**: estimated lines of code using the FP analysis;
- **E = 1.0769**: exponent derived from Scale Factors, previously calculated
- **EAF = 1.25**: product of all the Cost Drivers.

With this parameters we can compute the Effort value, that is equal to:
EFFORT = 2.94 * 5.428 $^{1.0769}$ * 1.25 **= 22.719 PM**

3.5 Duration, number of people and cost

From the estimated effort, we can deduce the project duration with the following formula:

DURATION = C * EFFORT$_{NS}$ $^{(D + 0.2 * (E - B))}$ * (SCED%/100) ,

Where:

- **C** is a constant with value 3.67;
- **EFFORT$_{NS}$** is the effort excluding the SCED multiplier;
- **D** is a constant with value 0.28;
- **E** is the exponend derived from Scale Factors previously found;
- **B** is a constant with value 0.91;
- **SCED** is a Cost Driver.

Therefore, we have:

DURATION = $3.67 * 22.719^{(0.28 + 0.2 * (1.0769 - 0.91))} = 3.67 * 22.719^{0.31338}$ = **9.766 M**

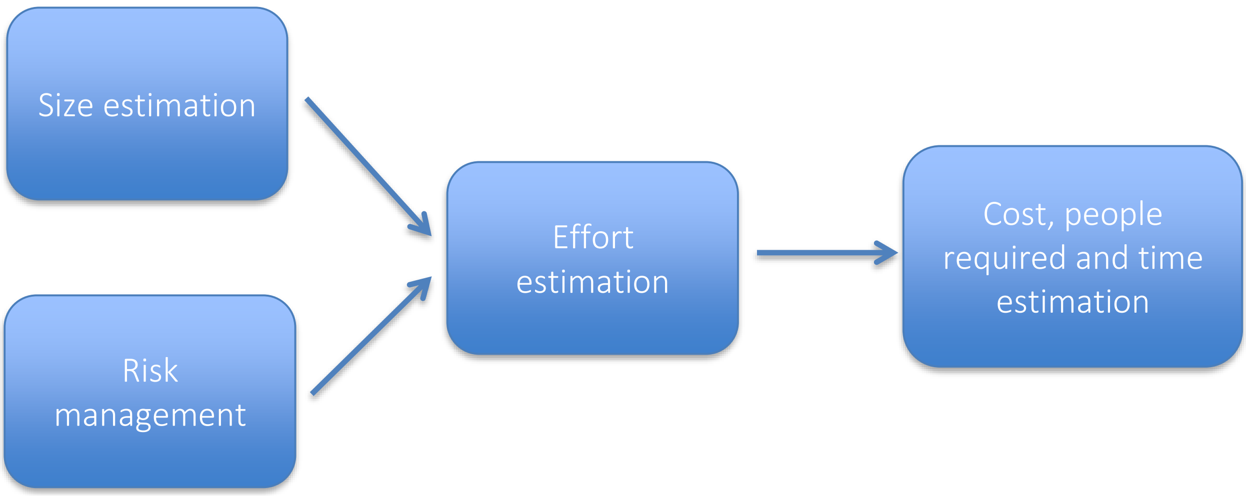
Now, from effort and duration, we can deduce the number of people needed to complete the project:

N$_{people}$ = EFFORT / DURATION = 22.719 / 9.766 = 2.326 = 2 people

Assuming a salary of $1500 per month per person, the total estimated project cost is:

COST = SALARY * EFFORT = 1500 * 22.719 = **$34078,50**

This is a recap of our estimation procedure:



3.6 COCOMO II online calculation

COCOMO II - Constructive Cost Model

Software Size Sizing Method: Function Points

Unadjusted Function Points: 118 Language: Java

Software Scale Drivers

Precedentedness	Low	Architecture / Risk Resolution	High	Process Maturity	Nominal
Development Flexibility	High	Team Cohesion	High		

Software Cost Drivers

Product

Required Software Reliability	Nominal
Data Base Size	High
Product Complexity	Nominal
Developed for Reusability	High
Documentation Match to Lifecycle Needs	High

Personnel

Analyst Capability	High
Programmer Capability	High
Personnel Continuity	Low
Application Experience	Low
Platform Experience	Nominal
Language and Toolset Experience	Nominal

Platform

Time Constraint	Nominal
Storage Constraint	Nominal
Platform Volatility	Nominal

Project

Use of Software Tools	Nominal
Multisite Development	High
Required Development Schedule	Nominal

Maintenance Off

Software Labor Rates

Cost per Person-Month (Dollars) 1500

Calculate

Results

Software Development (Elaboration and Construction)

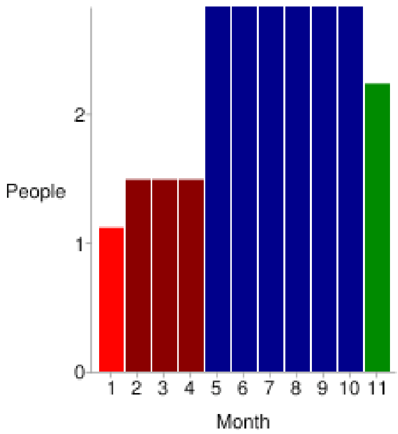
Effort = 24.6 Person-months
Schedule = 10.6 Months
Cost = $36849

Total Equivalent Size = 6254 SLOC

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	1.5	1.3	1.1	$2211
Elaboration	5.9	4.0	1.5	$8844
Construction	18.7	6.6	2.8	$28005
Transition	2.9	1.3	2.2	$4422

Staffing Profile



Software Effort Distribution for RUP/MBASE (Person-Months)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.2	0.7	1.9	0.4
Environment/CM	0.1	0.5	0.9	0.1
Requirements	0.6	1.1	1.5	0.1
Design	0.3	2.1	3.0	0.1
Implementation	0.1	0.8	6.3	0.6
Assessment	0.1	0.6	4.5	0.7
Deployment	0.0	0.2	0.6	0.9

Your output file is http://csse.usc.edu/tools/data/COCOMO_January_18_2017_14_56_20_626836.txt

Created by Ray Madachy at the Naval Postgraduate School. For more information contact him at rjmadach@nps.edu

4 Risks and recovery actions

4.1 Introduction

Risk management is an important phase of project planning, but it is often underestimated. Taking into account risks in advance means being proactive towards them. Primary objective of risk management is to avoid risks and to have a contingency plan in place to handle them in a controlled and effective manner.

There are some principles that we have always to follow:

- Encourage all stakeholders and users to point out risks at any time:
  It is important to have an open communication, the whole project can only benefit if no one try to hide information and risks;
- Modify identified risks as more becomes known and add new risks as better insight is achieved;
- Develop a shared product vision, because a shared vision by all stakeholders facilitates better risk identification and assessment.

In next paragraphs, we have analysed some risks, ranking them by probability and impact. We have then developed a contingency plan for those that are not negligible. Therefore, we are using a proactive risk strategy.

4.2 Risk Analysis

ID	*Type*	*Risk Description*	*Probability*	*Impact*
1	Project	Project size and complexity grow more than expected	High	Critical
2	Business	Customer not fully aware of project progress	High	Moderate
3	Business	Financial problems force reductions in the project budget	Moderate	Critical
4	Project	The development time is underestimated	Moderate	Critical
5	Project	Deadlines too close when drafting project documents	Moderate	Critical
6	Project	Key staff are ill at critical times in the project	Moderate	Serious
7	Project	Staff experience with used tools is not adequate	Moderate	Serious
8	Project	Changes to requirements that require major design rework are proposed	Moderate	Serious
9	Technical	The database used in the system cannot process as many transactions per second as expected	Moderate	Serious
10	Project	Integration tests shows that design of system has to be improved	Low	Critical
11	Technical	Car subsystem does not maintain an high level of reliability	Low	Critical
12	Technical	Database size grows more than expected	Low	Serious
13	Technical	An external handler changes its interface with the system	Low	Serious
14	Project	Not complete and correct understanding of requirements	Low	Critical
15	Techincal	Car internet access not always working, hard to make it reliable	Moderate	Critical
16	Technical	Bad choiche of car sensors	Low	Serious

4.3 Recovery Actions

ID	*Strategy*
1	Have a meeting with other team members, where have to be analysed what are the functions, components, interfaces or other piece of software which complexity is higher than expected. Then, check all the possible alternatives that try to simplify the design and the implementation. At the end, a document with all the taken design choices has to be drafted and has to be delivered to all the stakeholders involved.
3	Draft a briefing document for customers showing progress of the development, showing how the project is making a very important contribution to the goals of the business and presenting reasons why cuts to the project budget would not be cost effective.
4	Have a meeting with other team members, where point out problems during development that lengthened the schedule. Try to identify tools that would facilitate the development, or try to identify tasks that can be switched between team members. If the situation does not improve, take actions listed for risk 1.
5	Have a meeting with other team members, where point out what sections of document can be simplified in order to draft in time a simpler, but clear, document.
6	Try to identify tasks that can be switched between team members, what tasks can be delayed. If the situation does not improve, take actions listed for risk 4.
7	Have a meeting in which each team member explain what are key features that he knows for each tool. Try to identify documents, material or example on the web that can improve the knowledge.
8	Alert customer to potential difficulties and the possibility of delays because of changes in already approved RASD; investigate possibly buying-in or simplifying components to be developed.
9	Investigate the possibility of buying a higher-performance database; investigate the possibility to build more efficient algorithms, which interface with database.
15	Investigate the possibility to buy better internet modules for the cars; consider to make the cars send data to the server when internet connection is available again.

5 Project schedule and resource allocation

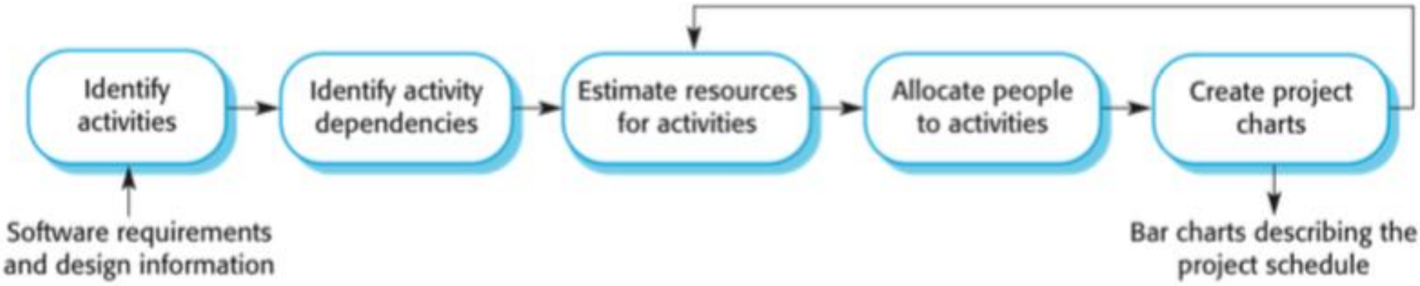
5.1 Introduction

This section describes how the work will be organised as separate tasks and when and how these tasks will be executed.

We should focus on:

- Splitting the project into tasks and estimating time and resources required for each of them;
- Organizing tasks concurrently and creating project charts.

The project scheduling process is summarized in the following diagram:



Some scheduling problems that we must keep in mind are:

- It's not easy to estimate the difficulty of a problem;
- Productivity is not proportional on the number of workers;
- Adding people to a late project makes the project late;
- Unexpected things always happen.

Therefore, the project development may differ from our schedule. For this reason, it is necessary to update the project schedule every certain amount of time to keep it consistent.

5.2 Tasks

In this section the project is breakdown into tasks. Each task is designed so that it should take about a week or two.

The tasks can be divided in the following categories:

- Requirements Tasks RT: relative to requirement analysis;
- Design Tasks DT: relative to software design;
- Setup Tasks ST: relative to tier setup (main server setup, car setup, database setup);

- Coding Tasks CT: relative to the project implementation and unit testing;
- Testing Tasks TT: relative to integration testing;
- Delivery Tasks DT: relative to the delivery of the project.

The dependencies between software components have been already analysed in the Integration Testing Plan Document. We can use them to decide the order of software components development and integration testing, taking advantage of the already found dependencies.

5.2.1 Requirements Tasks

Task ID	Task description
RT1	Analyse all the requirements and the specifications, make the RASD document

5.2.2 Design Tasks

Task ID	Task description
DT1	Study the solutions to our problem, develop the software architecture make the DD document

5.2.3 Setup Tasks

Task ID	Task description
ST1	Main server setup (correctly configured as a JEE server, Router class is working), database setup (database running, all tables created)
ST2	Car setup (correctly configured as a JEE server), phones setup (all phones in the equipment required section of the ITPD setup for development)

5.2.4 Coding Tasks

Task ID	Task description
CT1	Main server Map Controller
CT2	Main server Payment Controller
CT3	Main server Signup Controller
CT4	Main server Login Controller
CT5	Main server User Controller
CT6	Main server Car Controller
CT7	Main server Search Controller
CT8	Main server Reservation Controller
CT9	Main server Persistence Tier

CT10	Main server JSP
CT11	Main server Web Services
CT12	Car GPS, Seat Sensor, Dashboard Controller
CT13	Car Taximeter Controller, Car Info Controller
CT14	Car PEJControllerCar
CT15	App logic
CT16	App UI

5.2.5 Testing Tasks

Task ID	Task description
TT1	Search Controller → Car Controller
TT2	Search Controller → Map Controller
TT3	Signup Controller → Payment Controller
TT4	User Controller → Signup Controller
TT5	User Controller → Login Controller
TT6	Reservation Controller → Car Controller
TT7	Reservation Controller → User Controller
TT8	Reservation Controller → Payment Controller
TT9	PEJControllerCar Controller → GPS Controller, PEJControllerCar Controller → Seat Sensor Controller, PEJControllerCar Controller → Dashboard Controller
TT10	PEJControllerCar Controller → Taximeter Controller, PEJControllerCar Controller → Car Info Controller
TT11	Persistence Manager → Database
TT12	Business Logic Controllers → Persistence Manager
TT13	Business Logic Controllers → Car
TT14	Car→ Business Logic Controllers
TT15	Web Tier → Business Logic Controllers
TT16	Web App → Web Tier

5.2.6 Delivery Tasks

Task ID	Task description
DelT1	Deliver product

5.3 Task dependencies

Task ID	Effort [person-days]	Duration [days]	Dependencies
RT1	40	30	-
DT1	100	60	RT1
ST1	2	2	DT1
ST2	2	2	DT1
CT1	5	5	ST1
CT2	5	5	ST1
CT3	5	5	CT2
CT4	4	4	ST1
CT5	10	10	CT3, CT4
CT6	10	10	ST1
CT7	12	12	CT1, CT6
CT8	20	14	CT2, CT5, CT6
CT9	7	7	ST1
CT10	25	17	ST1
CT11	10	10	ST1
CT12	10	10	ST2
CT13	9	9	ST2
CT14	6	6	CT12, CT13
CT15	30	20	ST2
CT16	30	20	ST2
TT1	6	6	CT6, CT7 started
TT2	7	7	CT1, CT7 started
TT3	5	5	CT2, CT3 started
TT4	6	6	CT3, TT3, CT5 started
TT5	7	7	CT4, CT5 started
TT6	5	5	CT6, CT8 started
TT7	7	7	CT5, TT4, TT5, CT8 started
TT8	7	7	CT2, CT8 started
TT9	7	7	CT12, CT14 started
TT10	8	8	CT13, CT14 started
TT11	8	8	ST2, CT9 started
TT12	9	9	CT9, TT11, started a business controller
TT13	10	10	CT8, CT14
TT14	10	10	TT12, TT13
TT15	20	15	TT12, TT13, TT14
TT16	20	15	CT15, CT16, TT15
DelT1	10	10	TT16

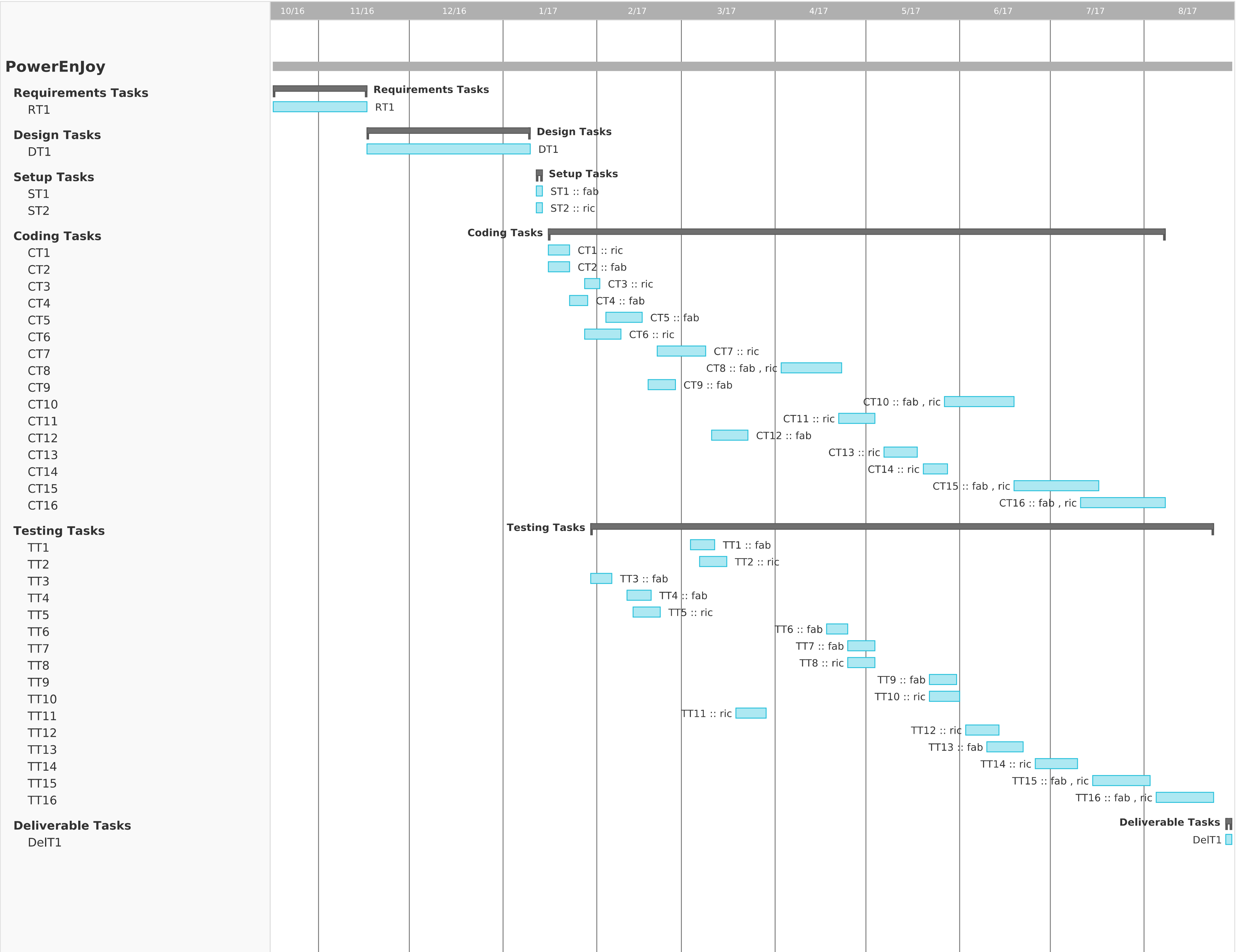
5.4 Resource allocation to tasks and charts

In order to allocate resources to tasks, the best thing to do is to find the critical path in the graph of the task dependences and allocate project members in a way to minimize interruptions on the critical path, according to the availability of each project member. We've chosen to alternate each other on the critical path.

However, due to the fact that we are just two people, there weren't a lot of opportunities to parallelize tasks.

We took into account personal availabilities and holidays.

In the following page there is the Gantt diagram that represents the project schedule along with resource (i.e. project members) allocation to tasks:


10/16
11/16
12/16
1/17
2/17
3/17
4/17
5/17
6/17
7/17
8/17
PowerEnJoy
Requirements Tasks
RT1
Design Tasks
DT1
Setup Tasks
ST1
ST2
Coding Tasks
CT1
CT2
CT3
CT4
CT5
CT6
CT7
CT8
CT9
CT10
CT11
CT12
CT13
CT14
CT15
CT16
Testing Tasks
TT1
TT2
TT3
TT4
TT5
TT6
TT7
TT8
TT9
TT10
TT11
TT12
TT13
TT14
TT15
TT16
Deliverable Tasks
DelT1
Requirements Tasks
RT1
Design Tasks
DT1
Setup Tasks
ST1 :: fab
ST2 :: ric
Coding Tasks
CT1 :: ric
CT2 :: fab
CT3 :: ric
CT4 :: fab
CT5 :: fab
CT6 :: ric
CT7 :: ric
CT8 :: fab , ric
CT9 :: fab
CT10 :: fab , ric
CT11 :: ric
CT12 :: fab
CT13 :: ric
CT14 :: ric
CT15 :: fab , ric
CT16 :: fab , ric
Testing Tasks
TT1 :: fab
TT2 :: ric
TT3 :: fab
TT4 :: fab
TT5 :: ric
TT6 :: fab
TT7 :: fab
TT8 :: ric
TT9 :: fab
TT10 :: ric
TT11 :: ric
TT12 :: ric
TT13 :: fab
TT14 :: ric
TT15 :: fab , ric
TT16 :: fab , ric
Deliverable Tasks
DelT1

6 Effort spent

We managed to distribute the workload fairly between days and team members in a way that allowed us to finish a few days before the deadline and have time for an accurate check in the last days.

The total amount of time required to build this document is about 15 hours for Riccardo Cattaneo and 13 hours for Fabio Chiusano.