Politecnico di Milano, A.Y. 2016/2017
M.Sc. Degree Programme in Computer Science and Engineering
Software Engineering 2 Project

Project Plan Document

Philippe Scorsolini,
Lorenzo Semeria,
Gabriele Vanoni

22nd January 2017

Contents

1 Introduction

1.1 Purpose and scope

The Project Plan document aims at investigating all costs and risks concerning the PowerEnJoy system development, giving a clear planning of the process.
We will proceed first of all with a size estimation through the Function Points technique, based on which we will estimate through COCOMO effort and cost of the development. These information will lead to a budget estimation, a resource allocation plan and an activities scheduling in the following sections.
We will then present a feasible split into atomic tasks, analysing their interdependencies. This will allow us, given the human resources allocated for this project, to produce a timetable assigning tasks and timeslots to the various components of the team.
We will finally give a brief analysis of risks, taking into account the probability and the relevance for each of them, presenting also a possible solution.

1.2 List of abbreviations

- FP: Function Points.
- ILF: Internal logic file
- ELF: External logic file.
- EI: External Input.
- EO: External Output.
- EQ: External Inquiries.
- LOC: Lines Of Code

1.3 List of reference documents

- COCOMO II Model Definition Manual, Version 2.1 (As found at http://bit.ly/1Bzg6T5, which points to a webpage of the Center for Systems and Software Engineering)

2 Estimations

This section will present an estimation of the expected size, cost and effort required for PowerEnJoy.
To estimate the size we will adopt the Function Points approach, taking into account all main functionalities, listed in the RASD, that the system has to offer to the various users. We will assume the system will be implemented using Java, so we will estimate the amount of the LOC needed to implement the business logic of our system, excluding the client parts.
For the cost and effort estimation we will make an Early Design analysis with COCOMO II, for which we present an in depth explanation of every Scale Factor and Effort Multiplier used.

2.1 Size estimation through Function Points

The Function Points estimation starts by extracting the main functionalities of the system, classifying them in a predefined set of classes and estimating their complexity accordingly to their specification found in the RASD. This estimation will be used then as basis for the COCOMO estimation.
The categories will be the following:

- Internal Logic Files
- External Interface Files
- External Inputs
- External Outputs
- External Inquiries

For each of the aforementioned categories we can have three level of complexity:

- Simple
- Medium
- Complex

By assigning,at each feature the system has to provide, a category and a complexity we will be able to compute the Unadjusted Function Points (UFP) value of the system as follows:

F := set of Function Types = {ILF, EIF, EI, EO, EIQ}
C := set of Complexities = {S, M, C}
$n\left[f,c\right]$:=number of functions of type f and complexity c
$w\left[f,c\right]$:=weight of function of type f and complexity c

$$UFC = \sum_{f \in F, c \in C} \left(n\left[f,c\right] * w\left[f,c\right]\right) \tag{1}$$

the weights we will consider will be the following ($w\left[f,c\right]$):

Function Types	Weights		
	Simple	Medium	Complex
ILF	7	10	15
EIF	5	7	10
EI	3	4	6
EO	4	5	7
EIQ	3	4	6

2.1.1 Internal Logic Files

Homogeneous set of data used and managed by the application. The application should manage an internal database, used to store the data about registered users, cars, power stations, reservations, drives, assistance tickets and operators.
Their complexity will be treated as Simple because all of them are trivially composed of a small number of fields.

ILF	Complexity	FP
Registered Users	Simple	7
Cars	Simple	7
Power Stations	Simple	7
Reservations	Simple	7
Drives	Simple	7
Assistance Tickets	Simple	7
Operators	Simple	7

2.1.2 External Interface Files

Homogeneous set of data used by the application but generated and maintained by other applications.
The system will check the driving licences using the APIs of the“Motorizzazione Civile” and the payment will be handled by an external Payment Handler, thus it will receive data about driving licence and payments by these two external components. Both of these will be treated as Average considering the fact that their structure will be trivial, but the interaction with external APIs will raise their overall complexity.
Assistance tickets will also be generated outside of the system boundaries and stored there. However, they will be treated as Simple due to their trivial nature.

EIF	Complexity	FP
Driving licences	Medium	7
Payments	Medium	7
Assistance tickets	Simple	5

2.1.3 External Inputs

Elementary operation to elaborate data coming from the external environment
The system will handle the interaction with users:

- Sign up: Medium complexity due to the necessary interaction with external components.
- Log in and out: operations of data retrieval from the DB, so their complexity will be Simple.

- Update user data: Medium as it could imply to check some data with external components again.
- Reservation of a car: Complex as it will address many entities of the system.
- Choice of the money saving option: Simple complexity as will only update one entity.

And with the Operators

- Update cars data: Simple complexity as it is just an entity update in the DB.

EI	Complexity	FP
Sign Up	Medium	4
Log In	Simple	3
Log Out	Simple	3
Update user data	Medium	4
Reservation of a car	Complex	6
Choice of the money saving option	Simple	3
Update cars data	Simple	3

2.1.4 External Outputs

Elementary operations that generate data from the external environment, usually includes the elaboration of data from logic files.
The System will generate the following External Outputs:

- Computed fairs: Medium as it will access some entities of the system and apply some elaboration to them.
- Money saving option enabled power stations: High as the system will have to access many entities and do complex elaborations on these data in order to return the power stations where the user that has enabled this option could plug the parked car.

EO	Complexity	FP
Computed fairs	Medium	5
Money saving Power stations	Complex	7

2.1.5 External Inquiries

Elementary operation that involves input and output, without significant elaboration of data from logic files.
The system shall be able to respond to information request by the user about:

- Free cars: Medium as it will possibly have to deal with many instances of the car entity.
- Their data: Medium due to the fact that it will deal with all the entities related to the user.

And by the Operators:

- Cars data: Medium because of the size of the data accessed.

- Assistance tickets: Medium because of the size of the data accessed.

EIQ	Complexity	FP
Free cars	Medium	4
User data	Medium	4
Cars data	Medium	4
Assistance tickets	Medium	4

2.1.6 Recap

So we have estimated as follow:

Function Types	Weights			Total Value
	Simple	Medium	Complex	
ILF	7	0	0	49
EIF	1	2	0	19
EI	4	2	1	26
EO	0	1	1	12
EIQ	0	4	0	16
Total UFP				122

2.2 COCOMO II

Cost estimation using COCOMO II aims at providing an objective cost estimation for a project. In this document we will use the Early Design model for the estimate.
As stated in the Manual[1], *«The Early Design model is a high-level model that is used to explore of architectural alternatives or incremental development strategies. This level of detail is consistent with the general level of information available and the general level of estimation accuracy needed.»*
Early Design, as the name suggests, has been developed for an early analysis of a project. For this reason in this document we will assume that none of the others has been written yet.

2.2.1 Introduction and Formulae

In the following paragraphs we introduce the formulae needed to complete the COCOMO cost estimation.

2.2.1.1 Calculating Person Month The amount of Person Month required for the given project is calculated using this formula:

$$PM = A \cdot Size^E \cdot \prod_{i=1}^{n} EM_i \tag{1}$$

Where:

- $A = 2.94$ by default. It may be calibrated on a company's local development environment as explained and suggested in the Manual (see Manual, Section 7).
- $Size$ can be expressed using either thousands of source lines of code (KSLOC) or unadjusted function points (UFP). The formula assumes that KSLOC are used but convertion between the two units is trivial since according to the Manual (Section 2.3), the SLOC/UFP ration for Java is 53. (*Note*: the table shows the SLOC to UFP ratio while the formula requires the KSLOC so the factor must be divided by 1000).
- E, the exponent of $Size$, is derived by the aggregation of 5*Scale Factors.*
- n is the number of Effort Multipliers (EM) and for Early Design is 7.
- EM_i are obtained by combining the Post-Architecture Design equivalent coefficients.

2.2.1.2 Calculating E The exponent E is used to take into account the economies and diseconomies of scale for the project. If $E < 1$ the project shows economies of scale, which means that the increment of effort needed grows with a factor smaller than the one representing the increase of project size.If $E = 1$, the economies and diseconomies are balanced. This factor is usually used for small projects. Finally, if $E > 1$ the project exhibits diseconomies of scale. This usually happens with big projects and is mainly due to increased intercommunication needs and greater integration overhead.

[1]The official COCOMO II Manual, as listed in Section 1.3 of this document.

To calculate E the following formula must be used:

$$E = B + 0.01 \sum_{j=1}^{5} SF_j \tag{2}$$

Where:

- $B = 0.91$ by default. It may be calibrated.
- SF_j are the 5 Scale Factors, as detailed in Table 10, Section 3.1 of the Manual. The 5 SFs are thoroughly analysed in the following parts.

2.2.1.3 Calculating PMAT: EPML EPML (Equivalent Process Maturity Level) is defined in the Manual and is based on the 18 Key Process Areas (KPAs) in the SEI Capability Maturity Model.

$$EPML = 5 \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} \frac{\%KPA_i}{100} \tag{3}$$

Where:

- n is the number of KPAs that was not assessed to be "Do Not Apply" – that is all Key Process Area that seemed applicable and relevant in our case.
- $\%KPA$ is the percentage of times that a specific goal is achieved, for each of the KPAs

2.2.2 Estimating Scale Factors

Scale Factors account for some important aspects regarding both the team and the project. They concur in the estimation of the exponent E, therefore having an exponential influence to the final result of our estimate.

2.2.2.1 PREC: Precedentedness This factor is high if the team is very experienced with the type of software that is to be developed.
The following table, extracted from the manual, provides some default features that help assessing the PREC factor value.

Feature	Low	Nominal/High	Very High
Organizational understanding of product objectives	General	**Considerable**	Thorough
Experience in working with related software systems	**Moderate**	Considerable	Extensive
Concurrent development of associated new hardware and operational procedures	Extensive	Moderate	**Some**
Need for innovative data processing architectures, algorithms	Considerable	Some	**Minimal**

According to the above table it seems reasonable to choose a PREC value of **Nominal (Somewhat Unprecedented)**, which translates in a numerical value of **3.72**

2.2.2.2 FLEX: Development Flexibility This factor is used to estimate the flexibility of the project with regards to the requirements and external specification.
The following table, extracted from the manual, provides some default features that help assessing the FLEX factor value.

Feature	Low	Nominal/High	Very High
Need for software conformance with preestablished requirements	**Full**	Considerable	Basic
Need for software conformance with external interface specifications	**Full**	Considerable	Basic
Combination of inflexibilities above with premium on early completion	High	Medium	**Low**

We have opted for **Full** (Low Flexibility) for the first two features since we will have to handle payments, which in turn require compliance with the banking system. It is reasonable to consider the bank's protocols to be very strict since they must ensure that payments are safe. This limits our flexibility in the development. A similar consideration can probably apply with the handling of our users' personal information, which must abide by strict laws. For the last feature a value of **Low** appears to be appropriate since no premium will be granted if the project is completed earlier.
This leads to a FLEX value of **Low (Occasional relaxation)**, which translates in a numerical value of **4.05.**

2.2.2.3 RESL: Architecture / Risk Resolution This Factor reflects the level of risk awareness and is also correlated with the risk analysis performed.

Feature	Very Low	Low	Nominal	High	Very High	Extra High
Risk Management Plan identifies all critical risk items, establishes milestones for resolving them by PDR or LCA.	None	Little	Some	Generally	**Basic**	Fully
Schedule, budget, and internal milestones through PDR or LCA compatible with Risk Management Plan.	None	Little	Some	**Generally**	Basic	Fully
Percent of development schedule devoted to establishing architecture, given general product objectives.	5	10	**17**	25	33	40
Percent of required top software architects available to project.	20	40	60	**80**	100	120
Tool support available for resolving risk items, developing and verifying architectural specs.	None	Little	Some	**Good**	Strong	Full
Level of uncertainty in key architecture drivers: mission, user interface, COTS, hardware, technology, performance.	Extreme	Significant	Considerable	Some	**Little**	Very Little
Number and criticality of risk items.	> 10 Critical	**5-10 Critical**	2-4 Critical	1 Critical	> 5 Non Critical	< 5 Non Critical

Taking into consideration all of the above table's features it seams reasonable to choose a RESL level of **High (Generally)**, which translates in a numerical value of **2.83**

2.2.2.4 TEAM: Team Cohesion Team Cohesion aims at estimating sources of turbulence caused by differences among stakeholders.

Feature	Very Low	Low	Nominal	High	Very High	Extra High
Consistency of stakeholder objectives and cultures	Little	Some	Basic	Considerable	**Strong**	Full
Ability, willingness of stakeholders to accommodate other stakeholders' objectives	Little	Some	Basic	Considerable	**Strong**	Full
Experience of stakeholders in operating as a team	None	Little	Little	Basic	Considerable	**Extensive**
Stakeholder team building to achieve shared vision and commitments	None	Little	Little	Basic	Considerable	**Extensive**

Taking into consideration that our small team has already worked on projects with optimal results and very good cohesion as shown in the above table, it seems reasonable to choose a TEAM level of **Very High (Highly Cooperative)**, which translates in a numerical value of **1.10**

2.2.2.5 PMAT: Process Maturity For this Factor, the Manual suggests to either use CMM levels where available or KPAs (Key Process Areas) as defined in the SEI Capability Maturity Model. Since our team has no CMM level, we have examined the KPA levels trying to assess the overall maturity of the team. According to the given Formula (see Formula 3 in this Section), since in our assessment 15 out of 18 KPAs were applicable (resulting in$n = 15$), we have estimated the **EPML** level to be **2.45**.
According to the table in the Manual, this value puts us in between of Nominal and High (CMM Level 2 and 3, respectively). It seems more cautious to choose a PMAT level of **Nominal (SW-CMM Level 2)**, which translates in a numerical value of **4.68**

2.2.2.6 Final Result Using the Scale Factors calculated in the previous paragraphs, we can calculate E as specified in Formula 2.

Scale Factor	Value
PREC	3.72
FLEX	4.05
RESL	2.83
TEAM	1.10
PMAT	4.68
Total	**16.38**

The final result, since $\sum_{j=1}^{5} SF_j = 16.38$ as calculated above, is

$$E = 1.0738 \tag{4}$$

2.2.3 Effort Multipliers

For Early Design, we need 7 Effort Multipliers that will concur in Equation 1. These multipliers are the result of the combination of one or more post-architecture **Cost Drivers**. They will be combined according to the Manual, as it will be better explained later (See subsection 2.2.4).

2.2.3.1 RELY: Software Reliability This parameter aims at weighting in the estimation the impact of a software failure, depending on the consequences it will have.

RELY Descriptors	Slight inconvenience	Low, easy, recoverable loss	Moderate, easily recoverable losses	High financial losses	Risk to human life
Rating levels	Very Low	Low	**Nominal**	High	Very High
Effort multipliers	1.34	1.15	**1.00**	0.88	0.76

It seems reasonable to choose a RELY level of **Nominal (moderate, easily recoverable losses)** since a failure will not be able to harm our users (the system can only unlock a car but has no control over the driving system). The most likely consequence of a software error will be the inability for a User to unlock a car or to terminate the rental. This will result in having to refund the client, have the operators take care of the single cases and maybe give the Users some form of discount for future rentals. Therefore, RELY will have a value of **1.00.**

2.2.3.2 DATA: Data Base Size This Cost Driver aims at taking into account the effort needed to create and maintain the test data that will be used during testing. Since we are creating this Driver in an Early Design estimate we can only assume the characteristics of the data we have to store. From the Assignments document we can infer that we will have at least the entities representing **Users** and **Cars**. Each User entry will likely store all the information in strings while the car's information that is actively user by the system is likely to be mainly numeric (position, battery percentage...). For the user we can assume 10 string fields that will hold a variable length of text, likely not exceeding 255 bytes. For the car, although the used information seem to be mainly numeric, we can assume that the database will also hold some of the car's information (like insurance number or frame number). Therefore we can assume few bytes for the foreseen numeric values and other fields that will likely not be more than 20 255-byte text fields. A small yet relevant test dataset would likely consist of approximately 100 users and 20 cars. Using these assumptions we can have an estimate of the final test database size that would be approximately 360kB. In order to take into account unforeseen tables and columns, it is sound to take an upper bound for the database of 2MB.
Therefore the Bytes/SLOC ratio is approximately **300**.

DATA Descriptors		$R < 10$	$10 \leq R < 100$	$100 \leq R < 1000$	$R \geq 1000$
Rating levels	Very Low	Low	Nominal	**High**	Very High
Effort multipliers	- -	0.90	1.00	**1.14**	1.28

From the above table, using the calculated Ratio (**R**) of 300, we choose a DATA level of **High** and a DATA value of **1.14**.

2.2.3.3 CPLX: Product Complexity Complexity is the result of many factors, illustrated in Table 19 of the Manual. For the sake of brevity we are not reporting the whole table here but just referring to the ratings provided there. For the **Control Operations** factor it seems appropriate to choose a level of Nominal since we are likely going to deal with distributed processing backed by middleware as well as some message passing. For the **Computational Operations** rating a level of at least Nominal is necessary since we will likely deal with statistical routines for the Money Saving Option (or, at the very least, we will need more than standard operations as described for level Low). Our software will probably not need complex **Device-dependent Operations** since it will only share basic information with the cars and the charging station. For this reason it is appropriate to choose a level of Very Low for this factor. As per **Data Management Operations** we decided to use a level of Low since it seems unlikely that complex database interaction will be needed. For **User Interface Management Operations** we opted for a level of Nominal since, although we will likely use maps in our application (which would suggest a level of Very High for the 2D graphic part), these more complex features will be provided by an external entity.
For these reasons, interpolating the above levels, the overall CPLX level is conservatively set to **Nominal** with a numeric value of **1.00**.

2.2.3.4 RUSE: Developed for Reusability Creating reusable software means putting more effort in every part of its development. In particular, the software itself must be more general, the documentation must be better written and more detailed (poses a constraint on **DOCU**), the testing must be more thorough (constraints **RELY**). For this reason it is strongly suggested to have a RELY of at least one level higher than the RUSE rating, while DOCU should be on the same level of RUSE or better (depending on the chosen RUSE level, for higher levels of RUSE it is suggested to choose equal DOCU levels).

RUSE Descriptors	None	Across project	Across program	Across product line	Across multiple product line
Rating levels	**Low**	Nominal	High	Very High	Extra High
Effort multipliers	**0.95**	1.00	1.07	1.15	1.24

Since we are not planning to reuse the software we are creating and it does not seem that there will be some across project reuse of components, it seems that a RUSE level of **Low (None)** is the most appropriate. Therefore, the numeric value of RUSE is **0.95**.

2.2.3.5 DOCU: Documentation Match to Life-Cycle Needs This Driver aims at estimating the thoroughness of the documentation with regards to the life cycles. Low or absent documentation will result in lower costs upfront but will probably make future changes and maintenance very expensive. For this reason a level of at least Nominal should ideally be chosen for every project. Since no other constraint is forced by the RUSE factor, a level of **Nominal (Right-sized to life-cycle needs)** seems to be the most fitting. This assigns a numerical value of **1.00** to DOCU.

2.2.3.6 TIME: Execution Time Constraint This Driver is used to estimate the computational demands of our software. It seems reasonable to choose an intermediate level for this parameter since there appears to be no need of complex computations. From time to time some non trivial computation will probably be necessary (e.g. for the Money Saving Option).

TIME Descriptors		<= 50% use of available execution time	70%	85%	95%
Rating levels	Low	Nominal	**High**	Very High	Extra High
Effort multipliers		1.00	**1.11**	1.29	1.63

For the reason stated above, we opted for a TIME level of **High** and a corresponding numerical value for TIME of**1.11**.

2.2.3.7 STOR: Main Storage Constraint Taking into account the increasing amounts of cheap and fast storage, since our software probably will not need huge amounts of storage data, it seems optimal to choose the lowest possible level for this Driver.
The lowest level for STOR is **Nominal**, with a numerical value of **1.00**.

2.2.3.8 PVOL: Platform Volatility This Driver aims at weighting the volatility of the“Platform” used. Platform is used as a generic term to refer to both hardware and software, depending on the specific case. In this project volatility may concern both, although it seems reasonable that the hardware will not need to be update as often as the software will.
Overall it seems appropriate to choose a **Nominal** level for PVOL, which translates into a value of **1.00**.

2.2.3.9 ACAP: Analyst Capability This Driver is used to evaluate the Analyst’s capabilities but not their experience. The main factors that are taken into account are analysis and design ability, efficiency and thoroughness, and the ability to communicate and cooperate. For this reason, although not experienced, we can assess the ACAP level for our team to be **Nominal** (55th percentile), which translates in a numerical value for ACAP of **1.00**.

2.2.3.10 PCAP: Programmer Capability This Driver aims at having an estimate of the programmers capabilities as a team. The main factors is the general ability to deal with non trivial COTS, the ability to cooperate, the thoroughness and the efficiency. As for ACAP, experience should not be a factor for the estimation of this driver’s level. For this reason, considered that we have already efficiently worked as a team and have used COTS, a **High** (75th percentile) level of PCAP is appropriate. This translates in a numerical value for PCAP of **0.88**.

2.2.3.11 PCON: Personnel Continuity This Driver is use to take into account the turnover of the personnel. Since we are not supposed to hire other developers and none of us is likely to leave the team, a **Very High** level of PCON is chosen. This translates in a numerical value of **0.81** for PCAP.

2.2.3.12 APEX: Applications Experience This Driver gives parameters to create an estimate for the team’s experience with this type of application. Since the experience with this specific type of software is none, the appropriate level is **Very Low** (Less than or Equal to 2 Months) which translates in a numerical value for APEX of **1.22**.

2.2.3.13 PLEX: Platform Experience Since we have limited expertise in the production use of many platforms (e.g. distributed middleware, databases), the most correct level for PLEX is **Very Low** (Less than or Equal to 2 Months) which translates in a numerical value for PLEX of**1.19**.

2.2.3.14 LTEX: Language and Tool Experience This driver is used to assess the team’s knowledge of languages needed to develop the program as well as the software needed for development, requirements analysis, program style and formatting and others. Since overall we have good expertise in some production languages (Java, Python and JS seem the most relevant for this project) and and average knowledge of many important tools used in production environments, it seems appropriate to choose a **Nominal** (1 year of experience) for this Driver. This means that a value of **1.00** will be used for LTEX

2.2.3.15 TOOL: Use of Software Tools This driver is used to estimate the Tools that will be used in the project (or that it will possible to use). Since we have already used widely adopted life-cycle integration tools we will set the TOOL level to **High**, which results in a numerical value of **0.90**.

2.2.3.16 SITE: Multisite Development This Driver is used to asses the team's distribution over the territory. If the components are physically separated and need to communicate using, for instance, group calls it will be harder to cooperate. Since our team is geographically located in the same area there will be plenty of occasions to work in the same room ("Fully collocated" in Table 33 of the Manual). Should we need to communicate in other occasions we will use any technology needed including interactive multimedia. For this reason we decide to use a SITE level of **Extra High** (Fully Collocated) and therefore a numerical value for SITE of **0.80**.

2.2.3.17 SCED: Required Development Schedule This Driver is used to describe the effect of schedule compression/expansion for the whole project. It is likely that at most few schedule stretch-outs will be required. However, it seems prudent to select a SCED level of **High** (130% of nominal schedule) to account for possible delays. The numerical value for SCED will therefore be **1.00**.

2.2.4 Early Design: Deriving the Effort Multipliers

As already stated, in order to use Equation 1 we need the Early Design Effort Multipliers. These are derived from the Effort Multipliers calculated in the previous subsection, combined as explained in section 3.2.2 of the Manual.
The following table shows which Post-architecture Effort Multipliers are combined to calculate the corresponding Early Design Effort Multiplier.

Early design Cost Driver	Post-architecture counterparts
PERS	ACAP, PCAP, PCON
RCPX	RELY, DATA, CPLX, DOCU
RUSE	RUSE
PDIF	TIME, STOR, PVOL
PREX	APEX, PLEX, LTEX
FCIL	TOOL, SITE
SCED	SCED

To create the combined values it is usually needed to sum the various levels using the following level-to-value mapping:

Level	Value
Very Low	1
Low	2
Nominal	3
High	4
Very High	5
Extra High	6

2.2.4.1 PERS: Personnel Capability This Effort multiplier is the combination of ACAP, PCAP and PCON. We have previously chosen a Nominal level for ACAP, a High level for PCAP and a level of Very High for PCON. Using the mapping shown in the beginning of the section, we have a combined result of **12**.

Descriptor							
Sum of Drivers	3, 4	5, 6	7,8	9	10,11	**12,13**	14, 15
Rating Level	Extra Low	Very Low	Low	Nominal	High	**Very High**	Extra High
Effort Multiplier	2.12	1.62	1.26	1.00	0.83	**0.63**	0.50

Our PERS Effort Multiplier will therefore be **0.63**

2.2.4.2 RCPX: Product Reliability and Complexity This Effort multiplier is the combination of RELY, DATA, CPLX and DOCU. We have chosen a Nominal level for all the multipliers but for DATA, that has been set to High. Using the mapping shown in the beginning of the section, we have a combined result of **13**.

Descriptor							
Sum of Drivers	5, 6	7, 8	9-11	12	**13-15**	16-18	19-21
Rating Level	Extra Low	Very Low	Low	Nominal	**High**	Very High	Extra High
Effort Multiplier	0.49	0.60	0.83	1.00	**1.33**	1.91	2.72

Our RCPX Effort Multiplier will therefore be **1.33**.

2.2.4.3 RUSE: Developed for Reusability This Driver will not be combined with others so we can use the same value calculated in section 2.2.3, which was **0.95**.

2.2.4.4 PDIF: Platform Difficulty This Effort multiplier is the combination of TIME, STOR and PVOL. We have chosen a Nominal level for STORE and PVOL, while the level for TIME is High. Using the mapping shown in the beginning of the section, we have a combined result of **10**.

Descriptor					
Sum of Drivers	8	9	**10-12**	13-15	16, 17
Rating Level	Low	Nominal	**High**	Very High	Extra High
Effort Multiplier	0.87	1.00	**1.29**	1.81	2.61

Our PDIF Effort Multiplier will therefore be **1.29**.

2.2.4.5 PREX: Personnel Experience This Effort multiplier is the combination of APEX, PLEX and LTEX. We have chosen a Very Low level for APEX and PLEX, while the level for LTEX is Nominal. Using the mapping shown in the beginning of the section, we have a combined result of **5**.

Descriptor							
Sum of Drivers	3, 4	**5, 6**	7, 8	9	10, 11	12, 13	14, 15
Rating Level	Extra Low	**Very Low**	Low	Nominal	High	Very High	Extra High
Effort Multiplier	1.59	**1.33**	1.22	1.00	0.87	0.74	0.62

Our PREX Effort Multiplier will therefore be **1.33**.

2.2.4.6 FCIL: Facilities This Effort multiplier is the combination of TOOL and SITE. We have chosen a levels of High and Extra High respectively for TOOL and SITE. Using the mapping shown in the beginning of the section, we have a combined result of **10** .

Descriptor							
Sum of Drivers	2	3	4, 5	6	7, 8	**9, 10**	11
Rating Level	Extra Low	Very Low	Low	Nominal	High	**Very High**	Extra High
Effort Multiplier	1.43	1.30	1.10	1.00	0.87	**0.73**	0.62

Our FCIL Effort Multiplier will therefore be **0.73**.

2.2.4.7 SCED: Required Development Schedule This Driver will not be combined with others so we can use the same value calculated in section 2.2.3, which was **1.00**.

2.2.5 Final calculations and PM estimations

Now we can compute the only part that was missing to calculate the result of Equation 1, which is $\prod_{i=1}^{n} EM_i$. The table below shows the Effort Multipliers and the final result for the product.

Effort Multiplier	Value
PERS	0.63
RCPX	1.33
RUSE	0.95
PDIF	1.29
PREX	1.33
FCIL	0.73
SCED	1.00
Result	**0.99**

Plugging all the values in the PM formula we obtain the following result:

$$PM = 21.6 \tag{5}$$

3 Project Tasks and Schedule

We have divided each phase of the project in activities. Fortunately many of them could, at least partially overlap. In this way it has been possible to carry forward them in parallel. We chose the classical approach of industrial engineering and operations research i.e. to represent the schedule with tasks against time using a Gantt diagram (in figure 1, until April 2017). The complete output of the planning tool GanttProject can be found in `gantt.pdf`.

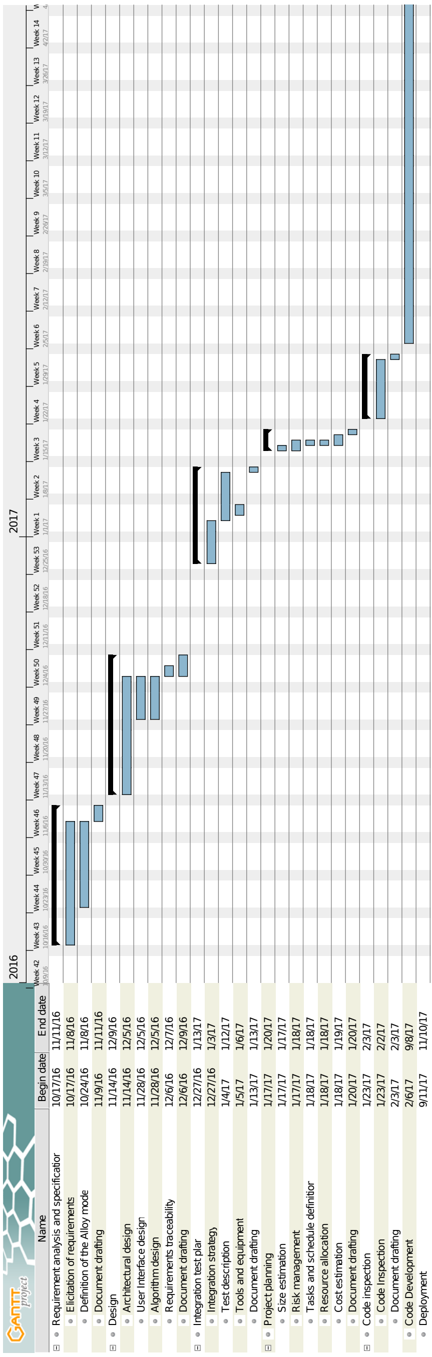

Figure 1: Gantt diagram of activieties and tasks

4 Resources and Task Allocation

The only human resources we have are ourselves. We have split the load of the hardest activities over more than one person, while the simplest were completed by a single member of the team. Percentages in the Gantt chart (figure 2, until April 2017) mean that the activity was split between different people in that fraction. The complete output of the planning tool GanttProject can be found in `gantt.pdf`.

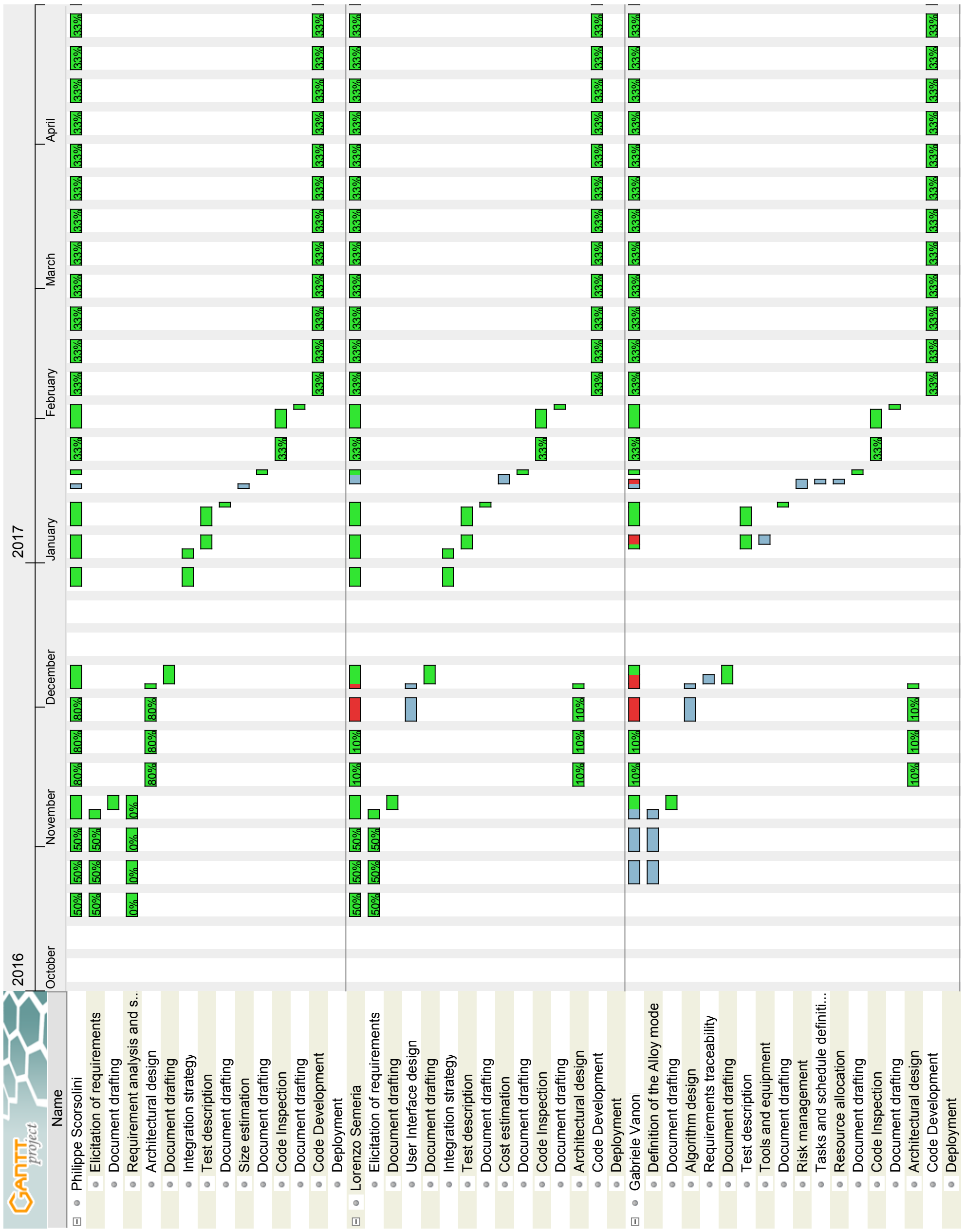

Figure 2: Gantt diagram of activieties and tasks

5 Project Risks

During the development of our project several things can go wrong i.e our project is subject to many risks. This section is aimed at analysing problems that can arise and at formulating strategies that can be put in practice to manage or prevent them.
Based on the literature on the subject we now list those that we think are the main possible risks.

R1) **Low budget**. Budgeting is done before the actual starting point of the project and expenditure items are forecasts. During the development of the project we realise that expenses are higher than budgeted.

R2) **Wrong schedule**. Project planning is done before the actual starting point of the project and schedules are estimates. During the development of the project we realise that some operations require more time than estimated.

R3) **Personnel shortfall**. In the project planning phase we estimate the amount of hours required to complete the overall work and, based on the deadlines and requirements, the number of people required and their competences. During the development of the project we realise that some operations require more people or different competences than estimated.

R4) **Requirement changes**. Requirements are set up in the RASD, but they are not set in stone. Stakeholders can change them or add new ones during all the phases of the development.

R5) **Bad external components**. Third party components can behave differently from what was expected when they were chosen in the design phase.

R6) **Performance issues**. After deployment it turns out that our hardware (servers, bandwidth, etc) can not support the load of requests.

R7) **Scalability issues**. After deployment and after many clients registered to our service, an upgrade of our hardware is needed. It turns out that our software does not scale on many server architectures, distributed database, etc.

R8) **Security issues**. After deployment a vulnerability is discovered in our software, so that user privacy is at risk, all our data are no more secure and also our physical machines (servers, cars, power stations).

R9) **Wrong user interface**. Feedbacks from users suggest that the user interface of our application is complex, not intuitive and difficult to use.

R10) **Regulation changes**. Governments continuously change laws and regulations. Our service in particular depends on regulations on driving licences, car sharing policies and civil insurance.

R11) **Bankruptcy**. The company could go bankrupt.

R12) **New competitors**. Car sharing market is an open market and new competitors can arrive at any moment.

R13) **Market extinction**. All markets may become extinct, if no user is not interested anymore in the service or product offered.

We are going to classify these risks according to their category, their probability (in a range Low, Moderate, High) and to their impact on the final outcome of the project. Furthermore for each problem we propose a strategy to manage or prevent it.

Risk	Probability	Impact	Strategy
Project risks			
R1	High	Critical	Try to make a good estimate to avoid the problem. However if it arises, ask managers to invest more since this is a core business project.
R2	High	Critical	Try to make a good estimate to avoid the problem. However if it arises, ask managers for more time or release a public beta with less functionalities but still usable.
R3	Low	Critical	Try to make a good estimate to avoid the problem. However if it arises, ask managers for the possibility of hiring new people.
R4	High	Marginal	Try to develop the system in such a way that it can be modified without totally rewriting it.
Technical risks			
R5	Moderate	Marginal	Try to avoid the problem choosing reliable third party vendors. However if it arises, change external software or if it is open source, adapt it to the need.
R6	Low	Marginal	Try to avoid the problem making performance testing. However if it arises, upgrade the hardware.
R7	Low	Critical	This problem has to be avoided in the design phase, since otherwise it is difficult to solve a fortiori. Software must be designed to scale!
R8	Moderate	Catastrophic	If a vulnerability is discovered in the software, disconnect everything from the Internet and fix it as soon as possible.
R9	Moderate	Critical	This problem can be avoided distributing in advance the application to some testers and asking them a feedback. However, if the problem arises, upgrade the user interface as soon as possible.
Business risks			
R10	Low	Critical	Often preliminary drafts of new laws are published before the approvation. The legal terms of the service have to be updated in this time frame.
R11	Low	Catastrophic	If the bankruptcy is not related to the project, there is nothing to do.
R12	Moderate	Critical	Develop quickly new features that the new competitors does not have and launch an aggressive marketing campaign.
R13	Low	Catastrophic	At this point there is nothing to do.

6 Effort spent

Component	Time spent (in hour)
Philippe Scorsolini	12
Lorenzo Semeria	16
Gabriele Vanoni	14