POLITECNICO
MILANO 1863

Project Plan Document for PowerEnJoy

Daniele Riva[*] Marco Sartini[†]

January 22, 2017

version 1.0

[*]matr. 875154
[†]matr. 877979

Contents

1 Introduction

1.1 Purpose

This document explains the Project Plan related to the *Power EnJoy* project. The aim of the document is to expose a first, estimated analysis about the imposingness, complexity, size and cost of the project, trying to provide a feasible (although ideal) schedule and work estimation. To provide a reliable estimation, we adopt the *Function Points* analysis to foresee the size of the project, and the *COCOMO II* model to foresee its effort and cost. In light of the obtained data, we propose a schedule for the project and a related resource allocation. In the end, will be reported a risks analysis and possible reactions to minimize their impact.

1.2 Scope

The project *Power EnJoy* is a platform based on mobile and web application thought to offer a car sharing service with electrical powered cars.

1.3 Definitions, acronyms, abbreviations

RASD requirements analysis and specifications document;

DD design document;

ITPD integration test plan document;

DBMS data base management system;

FP (FPs) function point(s);

UFP unadjusted function point;

(K)SLOC (kilo) source lines of code;

COCOMO II consctructive cost model II, model to estimate cost, effort and schedule of a project;

API application programming interface;

1.4 Reference documents

- RASD v1.0 available at https://github.com/marcosartini/PowerEnJoy/blob/master/releases/rasdPowerEnJoy.pdf
- DD v1.0 available at https://github.com/marcosartini/PowerEnJoy/blob/master/releases/dd.pdf
- ITPD v1.0 available at https://github.com/marcosartini/PowerEnJoy/blob/master/releases/itpd.pdf

Revision history

Name	Date	Reason For Changes	Version
Marco e Daniele	17/01/2017	Starting	1.0

2 Project size, cost and effort estimation

2.1 Size estimation: Function Points

UFP Complexity Weights

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

2.1.1 Internal Logic Files (ILFs)

The system is mainly centered on data, on both storing new ones and processing existing ones. It needs to store daily, hoping the great success of the service, a large amount of data regarding users, rentals, cars, and related ones to fulfill its duty. In particular, the Internal Logic Files are detect for:

- Users' information are stored by the system in a single table which contains `CF`, `Name`, `Surname`, `E-mail`, `Password`, `Driver license`, `Payment data`. It is a simple structure seen the data involved, mainly of type strings, and it suffices to adopt CF as identifier.

  This ends up to consider the structure as a low-weight one. [7 FPs]

- Cars information are stored by the system in a single table which contains `License plate`, `Battery level`, `Charging status`, `Seat number`. In addition it is stored the `Position` of the car as *latitude* and *longitude*. To complete the information, we also mark the car with a *state label*, supported by a state table (in a way, this last table behaviors like an enumerator table). All in all, it is a quite simple structure seen the data involved, and it suffices to adopt the License plate as identifier for the Cars, and the State name for the State table.

  This ends up to consider the structure as a low-weight one. [7 FPs]

- Maintenance and management employees constitute the service staff, and their information are stored by the system in two tables (one for each role) which contains of course the personal data such as `CF`, `Name`, `Surname`, `E-mail` and also the

`Password`. Since we prefer not to blend the various kinds of people, users and staff are not in the same table. But the inferences are the same as before: it is sufficient a single identifier and the structure appears simple.

  This ends up to consider the structures as a low-weight ones. [2 x 7 FPs]

- Power Grid Stations and Safe Areas are stored in a structure which considers the position and the extended human-readable address.

  This entities are pretty simples, then they can impact with a low cost value. [2 x 7 FPs]

- Rental and Reservation, are two entities which represent respectively a rental of a car and a reservation of a car; they store the user identifier and the car identifier as foreign keys, and of course the starting timestamp of the activity. For the Rental entity, furthermore, is stored also the end timestamp, the passenger number and the battery level spent (this last as computed value). The entity identifier is a multi-attribute formed by the user identifier, the car identifier and the starting timestamp.

  This ends up to consider the structures as average-weight ones. [2 x 10 FPs]

- The bill structure, as well, is based on three levels: the first contains the inherent bill data such as the amount, the date and the purpose; the second level contains service specific information reporting the user data, the car data and the timestamps; the third, finally, contains information about possibles discounts and overcharges.

  This ends up to consider the structures as an average-weight one. [10 FPs]

- The data structure thought to store an issue is pretty simple, containing as foreign keys the user who notifies the problem and the affected car; furthermore it is saved a description of the problem and its progress state.

  This ends up to consider the structure as average-weight one. [10 FPs]

- The last structure to be considered is the entity Keep aside, which stores the related rental (as foreign key), the starting timestamp and the duration. This is a low-weight entity. [7 FPs]

Internal Logic Files	**Complexity**	**FPs**
User Data	Low Cost	7
Cars	Low Cost	7
Maintenance employee Data	Low Cost	7
Management employee Data	Low Cost	7
Safe areas	Low Cost	7
Power Grid Station	Low Cost	7
Rental Data	Average Cost	10
Reservation Data	Average Cost	10
Bills	Average Cost	10
Issue	Average Cost	10
Keep aside Data	Low Cost	7
Total		**89**

2.1.2 External Interface Files (EIFs)

- In the Payment Handlers will be stored some piece of the payment data from our system.

  This provision to the cost is relatively low. [5 FPs]

- Maps service is enrolled to process data about addresses and geographical positions, in order to translate the human readable addresses in standard geographical positions. Data exchanged may be a huge, but all in all the complexity is low. [5 FPs]

External Interface Files	**Complexity**	**FPs**
Payment Handler	Low Cost	5
Maps service	Low Cost	5
Total		**10**

2.1.3 External Inputs (EIs)

- Login - Logout: these are two simple operations which require only some queries as select or update on the User entity. Hence the estimated contribute is *low*. [2 x 3 FPs];
- Register: it is an *average* complexity operation which involves only the user entity when adding a new one. It has however to check the correctness of the received data and if the user is already registered. [4 FPs];
- Start reservation: this operation is more complex than the previous ones, in fact it involves at least 3 entities (user, car and reservation) and need various steps to have

its job done (i.e. new reservation, update car and user states, trigger timeout). We mark as *highly*-cost this task. [6 FPs];

- Start rental is a complex operation; it involves many steps and entities, such as updating car and user's state and adding a rental instance. Many entities (car, user and rental) involved and different steps are the reasons to consider this procedure *highly*-cost. [6 FPs];
- Start keep aside: this is a quite complex operation which involves, other than the car, keep aside and rental entities, also the on-board car system. These reasons are enough we to consider *highly*-cost the operation. [6 FPs];
- Interrupt keep aside: it is a very complex operation because it involves more than 4 entities, that are keep aside, rental car payment and user. It also needs to relate with the on-board car system. *High* cost. [6 FPs];
- Notify issue: this simple operation interacts with the two entities issue and car, and with an easy step it updates the system state. *Low* cost. [3 FPs];
- Solve issue: it is a simple operation which involves at most two entities (issue and car) and through an easy step it updates the system state. *Low* cost. [3 FPs];
- End rental: this operation is complex because in order to complete its tasks it has to wait a notification by the on-board system and then it updates the user's state and the state of the car, and also sets the end time of the rental. The interactions are addressed to three entities and possibly also the the payment entity to compute the final price. It is clearly an *high-weight* cost operation. [6 FPs];
- End reservation: this is a complex operation which task consists in setting the end time of the reservation, checking if the timeout is expired, updating car and user states and finally delegating to the payment task or to the rental task manager. This hence results as an *high-weight* cost operation. [6 FPs];
- Edit settings involves a query to the settings table to update values such as the rental price and the keep-aside price. This is a very simple and straightforward operation, so it is counted as a low one. [3 FPs];
- Insert, delete and update Power Grid Stations and Safe Areas are simple operations which involves the modification of the register of Power Grid Stations and Safe Areas. They are not so complex, hence we consider them as low cost operations. [3 x 3 FPs];
- Insert, delete and update cars are simple operations which involves the modification of the register of the cars. They are not so complex, hence we consider them as low cost operations. [3 x 3 FPs];

External Inputs	Complexity	FPs
Login	Low Cost	3
Logout	Low Cost	3
Registration	Average Cost	4
Start Reservation	High Cost	6
Start Keep Aside	High Cost	6
Interrupt Keep Aside	High Cost	6
End Rental	High Cost	6
End Reservation	High Cost	6
Notify Issue	Low Cost	3
Solve Issue	Low Cost	3
Insert PG/SA	Low Cost	3
Delete PG/SA	Low Cost	3
Update PG/SA	Low Cost	3
Insert car	Low Cost	3
Delete car	Low Cost	3
Update car	Low Cost	3
Total		**64**

2.1.4 External Outputs (EOs)

- Generate password: it is a very simple method used to generate only a random password to be assigned to a user; it involves only the user entity and is made of two steps, for choosing the randoms characters and concatenating them. *Low* cost. [4 FPs]
- Compute bill (and also discounts): it is a very complex operation which involves many entities and claims several steps. In fact it computes both the rental and the keep aside costs (if any) and then it go looking for the discounts and overcharge to apply. Finally, it gathers the total price and notifies it to the external payment handler. Reasonably, this operation is quoted *highly*. [7 FPs];
- Generate unlock code: it is a simple method used by RentalManager to create and notify a code; the generated code is associated with rental and it is sent to the user and to the on-board system on the car for the future local matching. So it has only two short steps: to generate a random code and to notify it. It is a *low-weight* cost procedure. [4 FPs]

External Outputs	Complexity	FPs
Generate password	Low Cost	4
Compute bill	High Cost	7
Generate unlock code	Low Cost	4
Total		**15**

2.1.5 External Inquiries (EQs)

- Retrieve the list of available cars (position and other attributes): this operation involves different interaction and queries with entities to search for available cars in a certain area. The list is returned to the user in form of a nice map with the aid of the *GoogleMaps* API. This operation is not immediate as others, so it is an *average-weight* cost function. [4 FPs];
- View statistics: this operation needs a filter input to sift the wanted data aggregation. Of course it returns the requested data after an adequate process. Because of the aggregation presence, despite not ever will be executed the most loader query, in a pessimistic vision we tag the operation as an *high* cost one. [6 FPs]

External Inquiries	Complexity	FPs
Retrieve available cars	Average Cost	4
View statistics	High Cost	6
Total		**10**

2.1.6 Overall estimation

Starting from a recap of the identified values, we are ready to conclude the estimation about the number of lines of code.

Function type	Value
Internal Logic Files	89
External Interfaces Files	10
External Inputs	64
External Outputs	15
External Inquiries	10
Total	**188**

Assuming Java Enterprise Edition the platform to be adopted in the project development, and restricting the forecasts to the mere core application, without considering the presentation layer, we estimate a lower bound of

$$SLOC_{lower} = 188 \cdot 15 = 2820$$

an upper bound of

$$SLOC_{upper} = 188 \cdot 67 = 12596$$

and a likely average numbers of lines of

$$SLOC_{avg} = 188 \cdot 46 = 8648$$

where the coefficients are the parameters tagged to J2EE language from the *QSM analysis service* in the 5.0 version.

2.2 Cost and effort estimation: COCOMO II

This section reports an estimate of the costs and efforts which are supposed to be absorbed by the project. To obtain a reliable estimation, we adopt the COCOMO II model.

2.2.1 Scale drivers

COCOMO II bases its estimations on coefficients to be identified, for scale drivers, in this table:

Scale Factor values, SF_j, for COCOMO II Models

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	Level 1 Lower	Level 1 Upper	Level 2	Level 3	Level 4	Level 5
SF_j	7.80	6.24	4.68	3.12	1.56	0.00

Just have an overlook on the factors observed:

precedentedness represents the experience of the team in the development of such size project. We have only one project experience, and furthermore the level was appreciably lower. This quality is hence a *low* one. [4.96];

development flexibility represents the level of freedom with the respect to the requirements and the external specifications. The most of the requirements are restrictive, while on the hardware and architecture side we have a reasonable freedom. This quality is set to *nominal*. [3.04];

risk resolution represents the level of reactiveness to face with risks. We think to be at a *general* level, as also analyzed in the risks management section of this document. [2.83];

team cohesion represents the level of harmony among the team members on collaboration, cooperation, friendship and so on; it also takes into account the customer vision and concurrence. The developer team is close-knit, despite some timing issues. The customer is also smart and accommodating. We evaluate this quality as *high*. [2.19];

process maturity represents the level of performance achieved by an organization. Because we are planning the project, we employ skilled people, we involve the customer, we control the output produced and we try to prevent risks, we assume to be at a *nominal* level. [4.68]

The total scale driver value is 17.7.

2.2.2 Cost drivers

Cost estimation has to be made on a more granular analysis, that involves:

Required software reliability

For the company which will finance and maintain the service, the system is very important; anyway acceptable malfunctioning is tolerate and neutralized.

The level is hence treated as Nominal. [1.00]

RELY Cost Drivers						
RELY Descriptors	slightly inconvenience	easily recoverable losses	moderate recoverable losses	high financial loss	risk to human life	
Rating level	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

This estimation takes into account the size of the database on which the system will be tested on. At the moment, since the system is at design step, we infer to deal with a 4MB database, filled by a sufficient number of rows. [1.14]

DATA Cost Drivers						
DATA Descriptors		Testing DB bytes/pgm SLOC < 10	10≤D/P≤100	100≤/P≤1000	DP > 1000	
Rating level	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

In our intuition, this kind of system can be considered as *highly* complex, seen the activity of interest, the car sharing (which is substantially a new raising economical sector) and the users possibly target. [1.17]

CPLX Cost Driver						
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	0.73	0.87	1.00	1.17	1.34	1.74

Required re-usability

Reusability of the system is in a way limited to other realities with the same base concept. This means, for instance, an electric pedal assisted cycle sharing, where with some modifications can be adapted to the new business easily, as well as for electrical scooters or motorcycles. The alternatives are not so many, and it do not allows we to trust in a reusability greater than *high*. [1.07]

RUSE Cost Driver						
RUSE Descriptors		None	Across project	Across program	Across product line	Across multiple product lines
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	0.95	1.00	1.07	1.15	1.24

Documentation

The documentation is a key point to have a fully understandable product in the years to come, when the development will be over. Clearly we invest in a fine documentation. *Nominal* coefficient. [1.00]

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 level	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

This coefficient contributes for what concerns the CPU employing effort compared to the computational capabilities of the hardware. The burdensomeness of the system is considerable, inferring to deal with requests by users, cars, management, and also manipulating data. So about the 70% of the power is needed.

TIME Cost Driver						
TIME Descriptors			$\leq$ 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 level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	n/a	1.00	1.11	1.29	1.63

Storage constraint

This parameter takes into account the requested quantity of space where store information. Data are destined to grow daily due the physiological interaction among users and the system. In the long period this driver will assume an *high* value. [1.05]

STOR Cost Driver						
STOR Descriptors			$\leq$ 50% use of available storage	70% use of available storage	85% use of available storage	95% use of available storage
Rating level	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

Once started, the service and hence the main parts of the system will operate under the designed configuration for a long time.

It can however be expected that the client side of the system, affected by operating systems upgrades and evolution of devices, will call for improvements and updates periodically.

Currently, the major updates occur in average every five to six months. This last aspect contributes in the overall estimation stating as *nominal* the platform volatility. [1.00]

PVOL Cost Driver

PVOL Descriptors		Major change every 12 mo., minor change every 1 mo.	Major: 6mo; minor: 2wk.	Major: 2mo, minor: 1wk	Major: 2wk; minor: 2 days	
Rating level	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 capability of our analysts is not questioned, and we are sure that they will come up with a consistent solution. Reasonably, we assume to rely on a *high* cost driver value. [0.85]

ACAP Cost Driver						
ACAP Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating level	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

Our programmers are enough coordinated in teams, with good communication and cooperation qualities.

To be careful, however, we assume a *nominal* capability to be rated. [1.00]

PCAP Cost Driver						
PCAP Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.34	1.15	1.00	0.88	0.76	n/a

Application experience

This kind of application is not usually developed by our teams, and there are less than two months experience in the area, hence we can consider a *very low* cost driver. [1.22]

APEX Cost Driver						
APEX Descriptors	$\leq$ 2 months	6 months	1 year	3 years	6 years	

Rating level	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

In this part as well, we cannot boast a lot of experience, especially for the distributed architecture typical of *J2EE*. It is anyhow true that managing databases and user interfaces are skills acquired more than a year ago. It seems realistic to assume a *nominal* cost driver. [1.00]

PLEX Cost Driver						
PLEX Descriptors	$\leq$ 2 months	6 months	1 year	3 years	6 years	
Rating level	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 tools experience

Language and tools are well-known for more than two years, and in the last months this knowledge is highly improved.

Then this fact let us consider the cost driver as *high*. [0.91]

LTEX Cost Driver						
LTEX Descriptors	$\leq$ 2 months	6 months	1 year	3 years	6 years	
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.20	1.09	1.00	0.91	0.84	n/a

Personnel continuity

There is not a relevant turnover in the personnel, since the developer company is able to instill confidence in their employees and they are really loyal.

The numbers saying about a 5% per year turnover, so we take the *high* value. [0.90]

PCON Cost Driver						
PCON Descriptors	48% / year	24% / year	12% / year	6% / year	3% / year	
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.29	1.12	1.00	0.90	0.81	n/a

Use of software tools

As a forefront software company, we dispose of modern tools to assist all the phases of the development in a such integrated environment.

This cost driver is evaluated in *high* score. [0.90]

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 level	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 availability of the Internet service, exploited by the callings over IP and joined by the version control tools are the reasons which let us consider as *very high* the score of this cost driver. [0.86]

SITE Cost Driver						
SITE Collocation Descriptors SITE Communications Descriptors	International Some phone, mail	Multi-city and multi-company Individual phone, fax	Multi-city or multi-company Narrow band email	Same city or metro area Wideband electronic communication	Same building or complex Wideband elect. comm., occasional video conf.	Fully collocated Interactive multimedia
Rating level	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

Schedule constraints are not so strict, in fact the project is expected to take place in a quite environment, with no impressive deadlines.

We retain to deal with a *nominal* project. [1.00]

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 multipliers	1.43	1.14	1.00	1.00	1.00	n/a

2.2.3 Summary of cost drivers

We summarize the estimated driver coefficients to compute the final predictions.

Cost driver	**Factor**	**Value**
RELY	Nominal	1.00
DATA	Very High	1.14
CPLX	High	1.17
RUSE	High	1.07
DOCU	Nominal	1.00
TIME	High	1.11
STOR	High	1.05
PVOL	Nominal	1.00
ACAP	High	0.85
PCAP	Nominal	1.00
APEX	Very Low	1.22
PLEX	Nominal	1.00
LTEX	High	0.91
PCON	High	0.90
TOOL	High	0.90
SITE	Very High	0.86
SCED	Nominal	1.00
Total		**1.09343**

2.2.4 Effort equation

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

$$\mathit{effort} = A \cdot EAF \cdot KSLOC^{E}$$

where:

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

$$EAF = \text{product of all cost drivers} = 1.09343$$

$$\begin{aligned} E &= \text{exponent derived from the scale drivers} \\ &= B + 0.01 \cdot \sum_i SF[i] \\ &= B + 0.01 \cdot 17.7 \\ &= 0.91 + 0.177 = 1.087 \\ B &= 0.91 \text{for COCOMO II} \end{aligned}$$

With this parameters we can compute the effort value, which has a lower bound of:

$$\begin{aligned} \mathit{effort} &= A \cdot EAF \cdot KSLOC^E \\ &= 2.94 \cdot 1.09343 \cdot 2.820^{1.087} = 9.921PM \approx 10PM \end{aligned}$$

an upper bound of:

$$\begin{aligned} \mathit{effort} &= A \cdot EAF \cdot KSLOC^E \\ &= 2.94 \cdot 1.09343 \cdot 12.596^{1.087} = 50.47PM \approx 51PM \end{aligned}$$

and an average effort of:

$$\begin{aligned} \mathit{effort} &= A \cdot EAF \cdot KSLOC^E \\ &= 2.94 \cdot 1.09343 \cdot 8.648^{1.087} = 33.53PM \approx 34PM \end{aligned}$$

2.2.5 Schedule estimation

Regarding the final schedule, we are going to use the following formula:

$$duration = 3.67 \cdot \mathit{effort}^F$$

$$F = 0.28 + 0.2 \cdot (E - B) = 0.28 + 0.2 \cdot 0.177 = 0.3154$$

As a lower bound, we consider, with effort = 9.921 PM :

$$duration = 3.67 \cdot 9.921^{0.3154} = 7.568 months$$

while as an upper bound, we consider, with effort = 50.47 PM :

$$duration = 3.67 \cdot 50.47^{0.3154} = 12.64 months$$

and with an average effort of 33.53 PM we consider:

$$duration = 3.67 \cdot 33.53^{0.3154} = 11.111 months$$

which seem to be both reasonable estimates.

The estimations suggest to allocate to the project at least 2 people in the lower bounded case and at least 4 people in the upper bounded case, if the foreseen durations are acceptable to the customer and the management.

The actual team composition is two people. Let us take into account the *average* effort, to avoid being overly pessimistic. It comes out that the duration, in that case, would stretch to 16.765 months.

3 Schedule

The schedule chart in this section represents a possible time allocation, in a compact shape, just to stress the main activities to be considered. A deeper detail level is provided in the resources allocation, where the activities will be magnified. Since this document is made before the requirements analysis and the design of the system, it is not feasible to describe in details all the subparts of the activities, which will be rearranged during the completion of the architecture design document to point out the best schedule.

To be somehow reliable, we adopted the actual employed hours in the drafting of the documents to establish the possible length of those tasks. *Drafting* refers to both write and, more important, think what to write.

meeting with the customer involves the initial activities to pretty understand the whole project idea by the customer and get all the information needed for development;

project plan involves the activities devoted to the planning of the project and the estimation of its size, effort and cost;

requirements analysis involves the key activities devoted to the translation of the customer's idea into requirements and specifications of the project, bring to light limitations and solutions;

design of the systems involves the activities devoted to design the system and its sub components, complying to an architectural style;

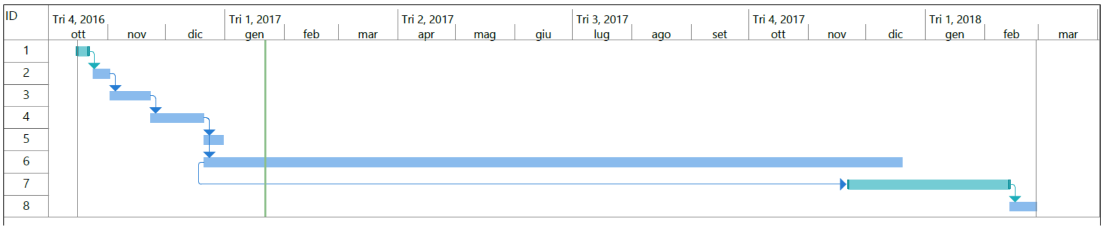
integration test plan involves the activities devoted to plan a feasible integration testing among the designed components;

development it is the most time consuming part of the project, when the real system will be given to birth. It is allowed to start as soon as the design document is completed;

integration testing involves the key activities devoted to the verification and validation of the integration among components of the system. It is better to start the actuation of the plan in the ending of the development phase;

release and deployment involves the activities devoted to the final release of the system to the customer and to the deployment to the users.

ID	Activity name	Duration	Starting	Ending
1	Meeting with the customers	6 days	16/10/2016	21/10/2016
2	Project plan	7 days	24/10/2016	01/11/2016
3	Requirement analysis	15 days	02/11/2016	22/11/2016
4	Design of the system	20 days	23/11/2016	20/12/2016
5	Integration Test Plan	8 days	21/12/2016	30/12/2016
6	Development	13 months	21/12/2016	19/12/2017
7	Integration testing	60 days	22/11/2017	12/02/2018
8	Release and deployment	10 days	14/02/2018	27/02/2018



4 Resource allocation

The schedule of the tasks above, to be fulfilled, needs a clear division of the activities and their sub-parts among the team members. This to properly take advantage of the parallelization inherent in a "team-structured" approach.

It is however fundamental to have the full team joined when accomplishing key activities. One of them is the *meeting with the customer*, which recovers the utmost role in understanding the aim and the objectives intended by the customer that have to be designed.

Once outlined together the possible requirements, structures, complexity of the overall project, it is time to produce a summary document, known as *Project Plan Document*, to show the project effort absorption, the estimated costs, the rough schedule, the first resource allocation and the risk management.

PPD	**Daniele**	**Marco**
1st week	FPs estimation	FPs estimation
	COCOMO II estimation	COCOMO II estimation
	Resource allocation	Schedule
	Risk management	Document drafting

As soon as the Project Plan Document is concluded and approved, the team can start analyzing the requirements of the project and deducing the specifications.

RASD	**Daniele**	**Marco**
1st week	Customer requests review	Customer requests review
	Goals outline	Scenarios outline
	Goals outline	Domain assumption outline
	Requirement outline	Use-case outline
2nd week	Feasibility evaluation	Feasibility evaluation
	UML outline	Function description
	Alloy world	Document drafting

As soon as the Requirement Analysis and Specifications Document is concluded and approved, the team can start sketching the architecture tailored to the system.

DD	**Daniele**	**Marco**
1st week	Main components analysis Components identification Deployment analysis	Main components analysis Components specification Interfaces identification
2nd week	Runtime analysis Runtime analysis	Algorithms identification UI mock-ups
3rd week	User Experience design Algorithms specification Requirements traceability	Deployment view Function description Document drafting

As soon as the Design Document is concluded and approved, the team can start producing a testing plan, specially aimed to the integration of the designed components.

ITPD	**Daniele**	**Marco**
1st week	Strategy definition Stubs and driver analysis Tests steps definition Supporting tools finding	Strategy definition Test data analysis Tests steps definition Document drafting

Slowly during the Integration Test Plan Document drafting is allowed to start the development step.

Development	Daniele	Marco
Month 1, 2	Deal external components Database interfaces Database modeling	Deal external components Database modeling Database connections
Month 3, 4, 5	Deal on-board system Structuring application logic Application logic Documentation	Deal on-board system External components interfacing Application logic Documentation
Month 6, 7, 8	Application logic Unit testing application logic Presentation layer modeling Mobile application modeling Documentation On-board interfacing	Application logic Web server set up Website modeling Website development Unit testing Code inspection
Month 9, 10, 11	Mobile application Code inspection Unit testing	Documentation Mobile application Unit testing
Month 12, 13	Application logic Algorithms improvement Integration tests Reviews	Communication protocols Integration tests Documentation Reviews

Approaching the end of the development period, let say no more than two months before it, the system should start being tested, complying the Integration Test Plan.

Integration testing	Daniele	Marco
1st month	Stubs development Driver development Core application tests	Testing database filling Testing data check Component tests
2nd month	External interactions tests Real world tests	User interface tests Real world tests

5 Risks management

5.1 Project risks

5.1.1 Team problem

The project will be developed by several people (programmers, engineers, project managers) who must interact and communicate to carry on the work. Being made up of people, a team may run into several problems.

Risk description	Feasible solution
Programmers tend to work alone, by implementing their own share without consulting the other programmers if they not finished own work. This can lead to various conflicts in functionality to implement or even also integration problems. *Probability: Low* *Effect: Slight*	To avoid this, the definition of each component and each function should be clear; also the work assignment for each team member must be precise and clearly defined. In Design Document these distinctions and definitions should be specified so clearly to avoid conflicts; Also another good countermeasure is planning several meetings during the implementation phase so that each component updates others on own work.
The abandonment (temporary or not) of one or more members of the project team (illness, accidents, resignation ...). *Probability: Low* *Effect: Catastrophic*	The solutions may be to replace the staff with new programmers (not very effective: Brook's law) or to reserve extra time at planning; so that if this problem occurs, the other team members can also finish the work of the absent member.

5.1.2 Changing requirements

Risk description	Feasible solution
It often happens that even the customers and stakeholders have completely clear idea of the application they want, so the requirements may change during the project development . You can not know in advance what requirements will want to change the clients *Probability: Moderate-High* *Effect: Moderate*	the only way to prevent this problem is to program with greater extensibility, so we'll be able to add new components and modify others avoid retouching the work already done.

5.1.3 Missed deadlines

Risk description	Feasible solution
During development it is possible that some components or features require more time than expected and therefore the risk is that some deadlines are not respected. *Probability: Moderate* *Effect: Moderate-Serious*	To avoid these delays are propagated with too much negative impact on the whole project, it may be assigned extra time at deadline and also plan a little support for those teams in trouble without cause delay to the development of other project parts.

5.1.4 Inefficient planning and early expiry

Risk description	Feasible solution
An unexpected problem can bring to a deadline advance or a bad planning can lead to not being ready in time. *Probability: Low-Moderate* *Effect: Serious*	To prevent this risk and to avoid a malfunction in the application , we need a good planning of each project process and begin with the development of the main features first (mobile app, application server) and fix the rest with next releases.

5.2 Implementation risks

5.2.1 Bad programing

Risk description	Feasible solution
Not always criteria for "good coding" are respected; for large-scale projects the code is often unreadable and poorly structured. *Probability: Moderate-High* *Effect: Moderate*	The solution may be to integrate the code with a good documentation and make it cleaner and understandable where possible.

5.2.2 Tests failures

Risk description	Feasible solution
During the testing phase can occur that the integration between the components do not provide the desired results, and so more time is needed to satisfy the conditions defined in test planning. *Probability: Moderate* *Effect: Serious*	Since this is a critical issue, components and resources of the project should focus on integration resolution so you can have the project functionality tested and working and then continue with the project development.

5.2.3 No scalability

Risk description	Feasible solution
A risk to be taken into account is the scalability of the system: the system should respond well to the increase of information to be managed (users, rentals, reservations) or you will have to intervene to re-define the parts that have not stood the increase. *Probability: Low* *Effect: Serious*	To avoid solving the problem with a modification to the system, we should make a correct estimate of the data size to be managed.

5.2.4 No flexibility nor extensibility

Risk description	Feasible solution
Another problem that can occur is the lack of flexibility and extensibility with which the project is developed. If the rush to meet the deadlines, you implement without thinking of a possible reuse of classes or other components, it will become very expensive to introduce new things. *Probability: Low-Moderate* *Effect: Serious*	To avoid this risk, it is necessary, in the RASD and in the DD, to specify as the components will have to be implemented so as to ensure a change without much difficulty.

5.2.5 Data loss

Risk description	Feasible solution
Because of some unexpected, it may happen that some of the project data will be lost during the development phase and lose so much of the work. *Probability: Low* *Effect: Catastrophic*	To prevent this risk, the optimal solution is to keep multiple backups of work in different places (both physical and cloud) in order to be ready for any kind of unexpected.

5.2.6 External services involvement

Risk description	Feasible solution
The application definitely depends on some external services (Google maps, localization, payment) and therefore experience problems in the moment that one of these services will not function properly. *Probability: Low* *Effect: Serious*	To guard against this risk, you need to choose the services that ensure the proper functioning and point to a good cooperation in order to resolve quickly any problems.

5.3 Economical risks

5.3.1 Rising of electrical energy prices

Risk description	**Feasible solution**
Assume that due to a particular event the prices of the production and distribution of the electrical energy will raise out of the rule, and this new cost is predict to stay high for a large number of years: if it will happen during the project development, it is possible that the customer will decide not to continue with the project. In fact, no more people will choose to drive the electrical cars, and the expected returns on investments will not take place. This situation, for an accurate manager, will end up with a project killing request. *Probability: Low* *Effect: Serious*	There is no an immediate solution to this risk, because the world is made by humans, and humans are weird people. To minimize the effects, it is feasible to revise the base problem changing the reference car fuel, and adapt the system to this new standard. The main concept of the system, however, will last.

5.3.2 Cities policies

Risk description	**Feasible solution**
In the cities where the project will be realized, the policies in matter of electrical cars stimulation could vary, taking a turn for the worst. *Probability: Moderate* *Effect: Slight*	To avoid being surprise of the events, it is important to maintain a close contact with the local governments.

6 Hours of work

Document	**Marco [h]**	**Daniele [h]**	**Total [h]**
Requirements and Specifications Document	48	49.5	97.5
Design Document	35	40	75
Integration Test Plan Document	26	22	48
Project Plan Document	20	20	40
Inspection Document	15	15	30
Overall revision	–	–	–
Total	144	146.5	**290.5**

7 References

To draw up this document, we refer to the sample Project Plan Document provided in the lectures.

QSM function points language table:
http://www.qsm.com/resources/function-point-languages-table