Project Plan Document

Enrico Migliorini, Alessandro Paglialonga, Simone Perriello

January 22, 2017

Contents

1 Introduction

1.1 Revision history

Table 1: **Revision History**

Version	**Date**	**Authors**	**Summary**
1.0	22/01/2017	E. Migliorini, S. Perriello, A.Paglialonga	First release

1.2 Purpose and scope

This document represents the Project Plan Document for PowerEnjoy. The main purpose of the document is to provide a methodological estimation of the project complexity, in order to provide a guidance for the definition of the required budget, the resources allocation and the schedule of the activities.

In section 2, we are going to use the Function Points (2.1) and COCOMO (2.2) approaches together to provide an estimate of the expected size of the System in terms of lines of code and of the cost/effort required to actually develop it.

In section 3, we will reuse these figures to propose a possible schedule for the project that covers all activities from the requirements identification to the implementation and testing activities.

Finally, in section 4 we will discuss the possible risks that the System could face during the various phase of the project and provide some general conclusions.

1.3 Definitions, Acronyms, Abbreviations

IFPUG: International Function Point User Group

1.4 References

- RASD, DD and IT Documents for the *PowerEnJoy* application
- Progressive Function Point Analysis Workbook
  available at `https://sourceforge.net/projects/functionpoints/`
- COCOMO II Model Manual
  available at `http://sunset.usc.edu/research/cocomoii/Docs/modelman.pdf`

2 Project size, cost and effort estimation

2.1 Size estimation function points

A function point is a conceptual measure that express the amount of business functionality a software provides, based on what the end user request and receives.

The Function Points approach, originally defined in 1979 by Allan Albrecht, provides an estimation of the size of a project. The approach takes as inputs the functional user requirements of the software and each one is categorized into one of five types:

- internal logic files (ILF)
- external logic files (ELF)
- external inputs (EI)
- external outputs (EO)
- external inquiries (EQ)

Once the function is identified and categorized into a type, it is then assessed for complexity (high, medium or low) and assigned a number of function points.

The 5 types of function points above, also known as Elementary Processes (EPs), can be grouped into 2 types of functions:

- Inputs, Outputs and Queries all qualify as Transactional Functions and,
- Internal Files and External Files are distinguished as Data Functions

These groupings are helpful in determining the types of elements that each function is broken down into, to determine the complexity of the EP and ultimately the number of function points that should be awarded for a given EP.

A Transactional Function is broken down into DETs and FTRs, while a Data Function is broken down into DETs and RETs.

- DET, Data Element Type is a unique user recognizable, non-repetitive field in all the kind of functions.

- FTR, File Type Referenced is a file type referenced by a transaction. An FTR must also be either an Internal or External file.
- RET Record Element Type is a user recognizable sub group of data elements within an Internal or External File.

So, basically, the elementary variables in functions are denoted as Data Element Type (DET). The functional complexity is computed as the total number of user identifiable groups that exists within DETs and is termed as Record Element Type (RET) in Data Functions and all referenced file types are counted as File Type Records (FTR) in Transactions Functions. A corresponding matrix holds the reference function point values for all function types (namely the ILF, EIF, EI, EO and EQ), with respect to the range of DET and RET/FTR in each function. These tables are shown in the following sections.

The following matrix represents instead the function points assigned for each subgroup of functions based on the corresponding complexity.

Table 2: **Function Point Weights**

Function Type	**Low**	**Average**	**High**
ILF	7	10	15
ELF	5	7	10
EI	3	4	6
EO	4	5	7
EI	3	4	6

2.1.1 Data functions

Data functions relate to the actions of storing and retrieving data in both local files or databases and external to the application through remote interfaces, associated middleware or outside the boundary of the application concerned.

Table 3 is used as a reference for deriving the complexity for both ILFs and ELFs.

Table 3: **Complexity matrix for EIF and EOF**

RETs	**DETs**		
	1-19	20-50	51+
1	L	L	A
2 to 5	L	A	H
6 or more	A	H	H

2.1.1.1 Internal Logic Files (ILFs)

The official IFPUG definition of an EIF is:

> An external interface file (EIF) is a user identifiable group of logically related data or control information referenced by the application, but maintained within the boundary of another application. The primary intent of an EIF is to hold data referenced through one or more elementary processes within the boundary of the application counted. This means an EIF counted for an application must be in an ILF in another application.

PowerEnjoy has to store information related to various kind of entities in order to provide the required functionalities. All the homogeneous information are stored into different files or tables in a database. To ensure that the entities are uniquely identified inside a specific file or table we assign to each one an ID, which is unique inside the file or the table. However, this is a system-generated sequence number, not an user-recognizable, so it does not count as a DET.

The first entities PowerEnjoy should track are surely the users; for this reason, we have a table for them storing for each one an email, a password, a social security number, a driving license number, a credit card number and a status (active/inactive). So, we can count 6 DETs, one for each of the data element identified above. Based on the information provided, we can thus judge that there will be only 1 RET, meaning that all 6 DETs will be seen by an user as a unique logically grouping.

Secondly, we have to store information about areas: GPS latitude, GPS longitude, city and parking slots. The System has to manage two different kind of areas: a parking area and a charging area, the latter being an extension of the former. There are various kind of strategies to manage this kind

of hierarchy at the data level; we choose to use the "single table strategy", in which the two classes of the hierarchy are mapped to a single table or file which has a discriminator column containing a value that identifies the subclass to which the instance represented by the record belongs. In our case, this discriminator is a boolean condition that is set to true if the specified record is a charging area. In addition to this field, for a charging area, we also have to add another field to store the number of charging slots associated to the charging area.
So, in the end we can come up with 1 RET and 6 DETs.

Another important piece of information is associated to the cars managed by the System. For each car, we have to store plate number and its status (available, unavailable, reserved or in use). Finally, we have to know in which area the car is; for this reason, we have a field that has the identifier of an area. Thus, for this kind of data, we came up with 1 RET and 3 DETs.

Closely related to a car we also need a table containing all the damages. We must note that a damage is autodetected from the car set of sensors and for these reasons some of the data associated to a specific damage belongs to the ELF category. Nevertheless, the System must store and maintain, for each damage, the identifier of the car on which the damage has been detected, two different timestamps related to when the damage was detected and when (optionally) it was solved. Of course, we have also a boolean flag that indicates if a damage has been solved or not. For this reason, we count 1 RET and 5 DETs.

For the main functionalities provided by our System we have to store other two different kind of homogeneous data: the first one for the reservations and the second one for the drivings. First of all, each reservation has a reference to both the ID of the user which has made the reservation and the ID of the car being reserved. Also, we store the time on which the reservation was made and the time on which it was concluded. Finally, we have a boolean flag to know if the reservation is currently active. We came up with 1 RET and 5 DETs.

The driving table is very similar to the previous one, storing a reference to both the ID of the user which driving the car, the ID of the car being driven, the time on which the drive started, the time on which it was concluded and the active flag. Apart from the previous fields, we have to store other information relevant for the evaluation of the fee applied to the user: three flag to know if the drive has to be applied a discount (the user has taken other passengers, the user left the auto with an high battery, the user plugged the car into a socket at the end of the ride) and other two for a surcharge (if the user left the auto with a low battery or if the car was left far from a charging area). In the end, we have 1 RET and 10 DETs.

Another homogeneous kind of data stored is related to the banking operations managed by our System. A banking operation can be related to an expired reservation or to a driving. For this reason, we have two optional data elements: the first one refers to the ID of a reservation, the last one to the ID of a driving. Of course we must also store the final fee of the specific banking operation, if it has been paid and if it has been processed. Thus, we have 1 RET and 5 DETs.

In the ILF we must surely count the various configuration files used by the System to define the amount of each banking operation. This kind of data relies on many different variables:

1. the fee per driving minutes contains how much a user should pay for each minute of drive
2. fee per expired reservation represents how much a user should pay for a reservation which is expired
3. passengers discount percentage represents the percentage of the discount to be applied in the case in which the user picked up other passengers during the drive
4. passengers number for discount is the data elements saying how many passengers a user should have picked up in order to qualify for the passengers discount
5. passengers time for discount is the data elements saying the minimum amount of time the passengers should be in the car in order to qualify for the passengers discount
6. high battery discount percentage represents the percentage of the discount to be applied in the case in which the user left the car with an high battery percentage at the end of the ride
7. high battery percentage for discount is the data elements saying the minimum battery percentage level requested to apply an high battery discount

8. plugged car discount percentage represents the percentage of the discount to be applied in the case in which the user connected the car plug to a socket of a charging area at the end of the ride

9. plugging car time indicates the maximum time the System waits until the user connects the plug, after which the discount is not applied anymore

10. away from charging area surcharge percentage represents the percentage of the surcharge to be applied in the case in which the user, at the end of the ride, left the Car away from a charging area

11. away from charging area meters represent how much is far away, id est the minimum distance between the car and the nearest charging area used to apply the surcharge at the previous point

12. low battery surcharge percentage represents the percentage of the surcharge to be applied in the case in which the user, at the end of the ride, left the Car with a low battery percentage

13. low battery percentage for surcharge represents the maximum percentage of the battery that has to be considered as low battery for the surcharge at the previous point.

We can easily conclude that we have 1 RET and 13 DETs.

Another kind of data we should manage is related to the configuration of other parameters related to the cars, namely:

1. locate car nearby range, id est the maximum range between an user and the available cars that the System should use;

2. unlock car nearby range, id est the maximum range between an user and the car he/she has reserved inside which he/she can unlock the car;

3. battery percentage available, id est the minimum percentage level that the System should use to classify a car as available.

We can easily came up with 1 RET and 3 DETs.

We must also store some kind of data related to the messages presented to the users:

1. registration email object contains the text of the object of the email that will be sent to an user asking to register to the System;
2. registration email subject, as before, but containing the subject;
3. reservation expired email subject contains the text of the subject of the email that will be sent to an user in the case of an expired reservation;
4. reservation expired email object, as before, but containing the object;
5. drive end email subject contains the text of the subject of the email that will be sent to an user at the end of the ride;
6. drive end email object, as before, but containing the object.

We count 1 RET and 6 DETs.

The System should also keep track of other two kind of informations: the users that are logged in the System and the users that have requested to register into the System. In order to accomplish the first objective, it has to store a uniquely identified token and the id corresponding to the logged user. We obviously have 1 RET and 2 DETs. For the second objective, it has to store a uniquely identified URLs and the id corresponding to the registering user. Even in this case we have 1 RET and 2 DETs.

At the end of the ILF analysis, basing on the value of the tables 2 and 3, we came up with the value indicated in the table 4.

Table 4: **ILF Weights**

Function Type	**Complexity**	**FPs**
User	Low	7
Car	Low	7
Area	Low	7
Damage	Low	7
Reservation	Low	7
Driving	Low	7
Banking	Low	7
Fees configuration	Low	7
Cars configuration	Low	7
Logged users	Low	7

Registering users	Low	7
Messages	Low	7
Total		84

2.1.1.2 External Logic Files (ELFs) Also called External Interface Files (EIFs), those are the data used and referenced by our System but not generated and maintained by it. The official IFPUG definition of an EIF is:

> An external interface file (EIF) is a user identifiable group of logically related data or control information referenced by the application, but maintained within the boundary of another application. The primary intent of an EIF is to hold data referenced through one or more elementary processes within the boundary of the application counted. This means an EIF counted for an application must be in an ILF in another application.

Assigning an FP value to an EIF is the same as assigning one to an ILF. First, determine the number of DETs and RETs in the ELF, then do a lookup in the table 3 to determine whether the ELF has a complexity of Low, Average, or High.

The main external data source the System should interact with is the set of cars. Each car has a set of sensors and the System must retrieve from them data about GPS latitude, GPS longitude, engine status, number of passengers, plugged status, door lock status and battery level. We must also note that each car can detect the presence of a damage through its sensors; for these reasons, the System should also retrieve data about a damage, namely if it is a major damage and an auto-generated text containing the description of the damage. We assume that each of these fields are maintained in a single RET in the car system. So, we came up with 1 RET and 9 DETs.

The second data source for the System is related to the communication with the GPS system of the mobile device. For the use of some functionalities, the user should provide its GPS position. For this reason, the System should retrieve the GPS latitude and longitude from the GPS system of the mobile device. This operation accounts for 1 RET and 2 DETs.

Another external source of data for our System is the mapping system, which is used for two different purposes:

- given an address, get the correspondent pair of GPS coordinates (reverse geo-coding);
- draw the graphical representation of different kind of maps.

While we can easily count 1 RET and 2 DETs for the first point, it is difficult to use the same kind of metrics for the second point. However, given the complexity of the interaction and the presumably high volume of data exchanged, it is reasonable to classify this kind of file as a complex one.

At the end of this analysis, using the values of tables 2 and 3, we came up with the following weights.

Table 5: **ELF Weights**

Function Type	**Complexity**	**FPs**
Car	Low	7
User GPS position	Low	7
Reverse geo-coding	Low	7
Graphical map retrieval	High	15
Total		36

2.1.2 Transactional Functions

Transaction functions relate to read/write operations performed on the data. The transaction functions will read or write data from and to an ILF or EIF. There are three basic type of Transaction functions which comprises of External Input, External Inquiry and External Output.

The official IFPUG definition of the previous functions is as follows:

> An external input (EI) is an elementary process that processes data or control information that comes from outside the application boundary. The primary intent of an EI is to maintain one or more ILFs and/or to alter the behaviour of the system. [...]
>
> An external output (EO) is an elementary process that sends data or control information outside the application boundary. The primary intent of an external output is to present information to a user through processing logic other than, or in addition to, the retrieval of data or control information . The processing logic must

contain at least one mathematical formula or calculation, create derived data maintain one or more ILFs or alter the behavior of the system.

An external inquiry (EQ) is an elementary process that sends data or control information outside the application boundary. The primary intent of an external inquiry is to present information to a user through the retrieval of data or control information from an ILF of EIF. The processing logic contains no mathematical formulas or calculations, and creates no derived data. No ILF is maintained during the processing, nor is the behavior of the system altered.

The IFPUG also provides a matrix for each of the previous group of functions. To note that the the matrix for EOs and EQs is the same.

Table 6: **Complexity matrix for EIs**

FTRs	**DETs**		
	1-4	5-15	16+
0-1	L	L	A
2	L	A	H
3 or more	A	H	H

Table 7: **Complexity matrix for EOs and EQs**

FTRs	**DETs**		
	1-5	6-19	20+
0-1	L	L	A
2-3	L	A	H
4 or more	A	H	H

To emphasize the normal flow of each function, we have decided to not treat each of them separately, but rather highlight the different kind of functions related to a specific interaction with the System.

Furthermore, in PowerEnjoy we basically have three different kind of clients: the web client, the mobile device and the car board. These are the primary sources of input for the System and for this reason we decided to classify all the provided functionalities basing on this distinction.

Web clients The web clients are used by users of the System to perform three kind of operations.

The first operation is a registration. The user that wants to access the functionalities of our System should insert their email, SSN, driving license number and credit card number. During the process, we can safely predict that all of the data will be "transacted" to a single record in an ILF or FTR, namely the ILF corresponding to the users. For this reason, we have 1 FTR and 4 DETs for this transactional function. We have to add 1 DET for the submit operation, which is the function activated by the user (likely through a button) to send its data. The whole operation is internal, so belongs to the EI category. We count an average complexity corresponding to 4 FPs.

However, during this function, the System generates a random URL to be sent to the email the user provided through the external mailing system. So, in this process, the System

- Generates a random and unique URL. We can assume that all the input data will be used to generate the URL. The URL must then be stored into an ILF associated to the corresponding user. This accounts for an EO process with 1 FTR and 4 DETs, corresponding to a low complexity and 4 FPs.

- Ask the external System to send the mail to the user email address. This process reference both the user ILF to retrieve the email address and the message ILF to retrieve the subject and the object of the mail. Thus, it accounts for an EQ process with 2 FTRs and 3 DETs, corresponding to a low complexity and 3 FPs.

So, the total for the registration operation amounts to 11 FPs.

The second operation is the confirmation of a registration through an URL. After the external mailing system send the email to the user, the latter can confirm the activation using this URL. This is a completely internal operation (an EI process) that access the ILF containing the user corresponding to that URL and modifies the ILF related to the user itself. It accounts for 2 FTRs and 1 DET, resulting in a low complexity and 3 FPs.

2.1.2.1 Mobile device Mobile devices are also used by users to interact with the System.

The first kind of operation performed with a mobile client is a login operation. The first part of this operation is an EI process: the user enters his/her email and password and ask to log in the System. It accounts for 1 FTP and 3 DETs (email, password and log in function) and can be categorized with a low complexity and 3 FPs. However, in the same operation, the System must generate, store and send back to the user a security token: this can be categorized as an EO process. We can assume that all the users input data are used to generate the random token; this accounts for 1 FTR and 4 DETs, resulting in a low complexity output corresponding to 4 FPs. The total for the login operation is of 7 FPs.

Thus, for each of the following transactions, the mobile device must send the previously retrieved token to the server. This operation involves the retrieval of the user data associated to the token, so it involves the ILF that tracks the pair of tokens and users. It is a completely internal operation, id est an EI process, that counts 1 FTR and 1 DET.

After a successful login, an user can retrieve the list of all the cars near him/her. In order to accomplish this goal, he/she can decide to either give a specific address or his/her GPS coordinates and then submit the data. In the first case, there is an EI process includes including the insertion of an address and the selection of the "locate cars" function. It accounts for 0 FTR (we do not maintain any ILF/ELF for this input) and 2 DETs.

The submission of the address trigger an EQ process, which asks the external mapping system the reverse geo-coding. In a similar manner to what we have previously done, we can assume that this operation involves 1 FTR and 2 DETs.

Then, the pair of GPS coordinates is sent to the System, which elaborates the data and returns to the user the list of cars near him/her whose status is available. We can see how this is an EO operation, because it also involves some calculation to see the cars near the user. In order to fullfil this transaction, the System should access the ILF containing a parameter for the maximum distance between the user and the car, then access another ILF containing all the cars, selecting only those whose status is available and near the user and presenting all their informations. So, we have 2 FTRs and 7 DETs for this EO operation.

Lastly, the System asks the external mapping system to give a graphical representation of the previous list of cars on a map. This is an EQ process which involves an high quantity of data, thus accounting for an high complexity and 6 FPs.

In the end, the retrieval of the list of cars through an address is a process characterized by:

- an EI process involving the insertion of an address, the selection of the appropriate function and the sending of an authentication token: 2 FTRs, 3 DETs; Low Complexity; 3 FPs;
- an EQ process involving 1 FTR and 2 DETs; Low Complexity; 3 FPs;
- an EQ process involving a High Complexity operation; 6 FPs;
- an EO process involving 2 FTRs and 7 DETs; Average Complexity; 5 FPs.

The total for this process is of 17 FPs.

If, on the other hand, the user decides to directly send the pair of his/her GPS coordinates, the process is slightly different. The EI process does not include an insertion of a text, but only the selection of a function; so, 0 FTR and 1 DET.

There is another kind of EQ process, which asks the GPS system of the device to get the pair of GPS coordinates of the user. It accounts for 1 FTR and 2 DETs.

Then, the pair of GPS coordinates is sent to the System for the elaboration, in a similar manner to what we have seen in the previous point.

So, the only difference is that we have only 1 DET less for an EI process, which does not modify its complexity. So, the total count for this process is again of 17 FPs.

Another kind of function selectable by an user is the retrieval of all the areas in a given city. This transaction which involves:

- an EI process in which a user insert the name of a city and the specific function: 0 FTR, 2 DETs; Low Complexity; 3 FPs;
- an EO process in which the System retrieve all the areas belonging to the given city accessing the appropriate ILF: 1 FTR, 8 DETs; Low Complexity; 3 FPs;
- an EQ operation in which the System asks the external mapping system to display the list of areas on a map: we categorize it as an operations which involves large quantities of data and so it is, presumably from

different FTRs, and for this reason we assign to it an high complexity and 6 FPs.

The total for this transaction is 12 FPs.

When users are logged into the System, they can also decide to reserve a car from the list of the available cars shown at the end of the previous transaction. The first part is an EI process, in which the user simply selects a car from the list. The selection triggers an action that sends, besides the token, the id of the car which the user wants to reserve. So, we have 2 FTPs and 2 DETs, resulting in a low complexity and 3 FPs.

During the same transaction, the System has to display a message to the user have to check, basing on the user and the car ids, that the user has no active reservation or driving and similarly that the car has not yet been reserved or driven and that it is still available. So, it has to access three ILFs containing the informations related to this data: the one for the reservation, providing it the user id and car id; the one for the driving, providing it the user id and car id again; the one for the car, checking that the status of the car is setted to available. Then, if the reservation can be made, the System has to update the ILF corresponding to the reservation (inserting a new reservation) and the one corresponding to the car (changing its status to reserved). We only count the same ILF and DET once, but during the insertion of the new reservation the System has to store also the data related to the start time and setting the active flag to true. Thus we come up with 3 FTRs and 7 DETs, resulting in an average complexity and 5 FPs.

The total for this transaction is of 8 FPs.

If the user does not unlock a car in a one hour time, the System should also charge the user with a fee, performing an EO process in which the user will be notified with an email containing the details of the expired reservation. This process involve the external mailing system, during which the System will access the mailing ILF (2 DETs), the reservation ILF (3 DETs) and the banking ILF (2 DETs). This will result in 3 FTRs and 6 DETs, accounting for 5 FPs.

Through his/her device, the user can also unlock a previously reserved car. In the first step, the user select from his device the function to unlock the car. This will send to the System his/her GPS coordinates, the reservation id and the token. This is an EI operation of 2 FTR and 4 DETs, contributing with 3 FPs.

The selection of the unlock function results in an EQ to the GPS system of the device, which will be called to retrieve the user GPS pair of coordinates; we count 1 FTR and 2 DETs for this process. For this reason, it contributes with 3 FPs.

In order to display a proper message to the user, the System should access 2 ILFs to evaluate if he/she can be authorized to unlock the car:

- the ILF file containing the parameter of the maximum distance range between an user requesting the unlock and the car to be unlocked
- the ILF file containing the GPS longitude and latitude of the car

So, this EO operation accounts for 2 FTRs and 3 DETs. It contributes for 4 FPs.

Laslty, the System will perform an EQ process, sending to the appropriate car a piece of data to unlock the car. It is safe to assume that this data will only have 1 DET and will not be stored in an ILF file; the result is a low complexity and 3 FPs.

The whole process contributes with 13 FPs.

The last kind of operation that an user can perform through the mobile device is the selection of the money saving option. When the user wants to select this option, he/she has to enter a destination address and submit the data. It accounts for an EI process with 0 FTR and 2 DETs, id est a low complexity process contributing with 3 FPs.

The System ask the reverse geo-coding of this data to the GPS System of the module. Again, it involves 1 FTR and 2 DETs, resulting in an EQ low complexity process which contributes for 3 FPs.

Then, the System has to display a result to the user. For this reason, it access the ILF files containing the informations of the area, basing on their city, GPS latitude, GPS longitude, free charging slots and free parking slots. So, this is an EO process with 1 FTP and 5 DETs, resulting in a low complexity and 4 FPs.

The total for this operation is of 10 FPs.

2.1.2.2 Car board The last kind of client for our System is the car board, which will be installed on each car. The board have an internal system which continuously fetch the data coming from the car sensors.

Thus, there is a large number of EI processes, involving a big amount of data exchanged between the car and the System to update the data of a drive. For this reason, we can assume that this process has an high complexity, resulting in 15 FPs.

During the ride, the System should also display the current fee to the user. This is an EO operation which can be classified as having a low complexity, resulting in 4 FPs.

At the end of the ride, the System should then charge the user with a fee. In order to do so, it has to evaluate the data both fetched during the drive and at the end of the ride. Even in this case, the operation involves a large amount of data to be processed, among which the update of various ILFs (mainly those related to the cars and to the banking operations) and the processing of ELF data of the car. It is safe to assume that this EO process has an high complexity, contributing for 7 FPs.

Then, it has to invoke the external banking system in order to charge the user with the final fee. This operation does not involve any mathematical formula or calculation, but only the retrieval of the data from the ILF containing the banking information to retrieve the final fee and the credit card of the user to charge. This EQ process results in 1 FTP and 2 DETs, resulting in a low complexity and contributing for 3 FPs.

Lastly, it has to perform an EQ process to send an email to the user with all the details of his/her drive, together with the final fee. For this scope, it has to access the mailing ILF (2 DETs), the driving ILF (7 DET) and the banking ILF (1 DET), resulting in 3 FTRs and 10 DETs and accounting for 4 FPs.

2.1.3 Overall estimation

Table 8: **Function Points overall estimation**

Function Points	**Value**
ILF	84
ELF	36
EI	43
EO	33
EQ	40
Total	***236***

2.2 Cost and effort estimation COCOMO II

2.2.1 Brief introduction to COCOMO II

COCOMO (COnstructive COst MOdel) is a cost estimation algorithm allowing to estimate the time, effort and money needed to develop a project. It provides an empirical non-linear model based on two series of values:

- **Scale Factors**, providing a gross estimate of the effort needed for the project.
- **Cost Drivers**, which can be *Early Design* or *Post-Architecture* and are applied to the effort as a multiplier.

COCOMO was originally published in 1981, based on a study of 63 projects of varying size and languages. COCOMO II, published in 2000, extends COCOMO by savoiding several underlying assumptions, such as waterfall development and stable requirements, and separating *Early Design* from *Post-Architecture* effort multipliers.

The general formula for calculating the needed Person-Months is

$$PM = 2.94 \cdot Size^E \cdot \prod_{i=1}^{n} EM_i$$

where

$$E = 0.91 + 0.01 \cdot \sum_{i=1}^{5} ScaleFactor_i$$

and size is expressed in Kilo Source Lines of Code, possibly derived from Function Points.

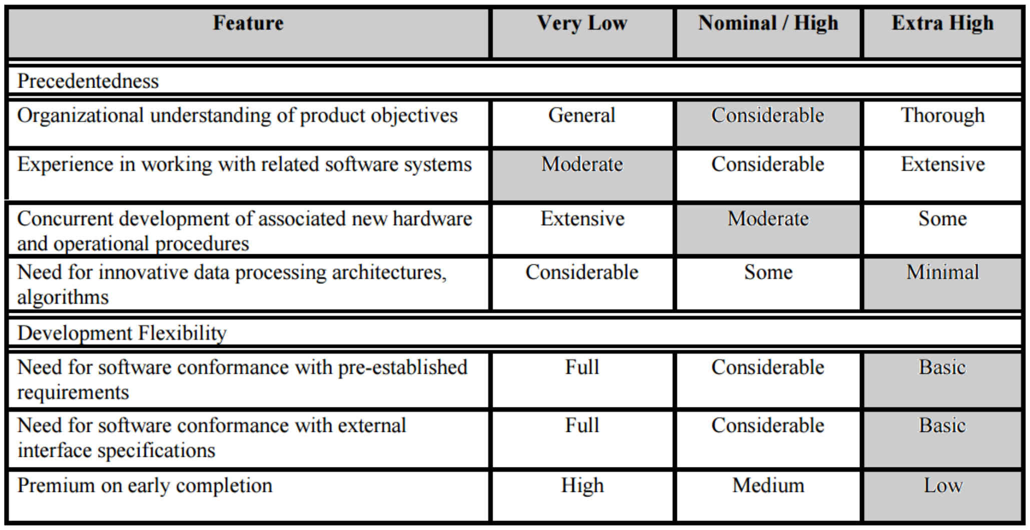
This estimate is made after planning the architecture, so we will use the *Post-Architecture* model.

2.2.2 Scale Factors

On the tables are marked in grey the values we believe appropriate.

Precedentedness: Low We have never built any similar application, although the structure of the application is not different from other projects that we have worked on. Between the three of us, we have experience of Server-Client architectures, Database Management, persistence in web applications, and an extensive knowledge of algorithms.

Development Flexibility: High Management did not impose a choice of framework, architecture or any set of constraints except the product goals. We were therefore free to choose how to build our project.



Feature	Very Low	Nominal / High	Extra High
Precedentedness			
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]
Development Flexibility			
Need for software conformance with pre-established requirements	Full	Considerable	[✓ Basic]
Need for software conformance with external interface specifications	Full	Considerable	[✓ Basic]
Premium on early completion	High	Medium	[✓ Low]

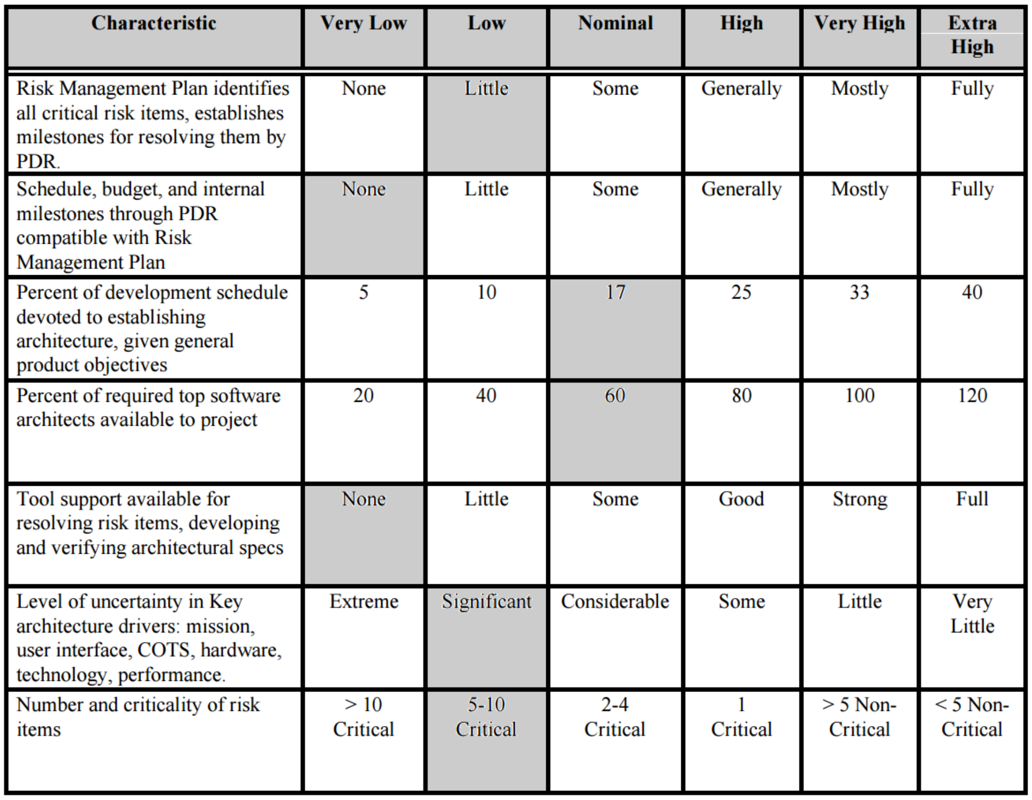
Figure 1: PREC and FLEX checklist

Architecture/Risk Resolution: Low We were not provided with little information about risk management, budget and architecture, and had to work it out ourselves. We were allowed a high degree of freedom, and the company intervened into planning and periodic check, but we were provided little detail about the practical implementation of the preferred architectures and risk management.

Team Cohesion: High We had a couple of small misunderstandings early on, but managed to work them out and have been efficiently working together.

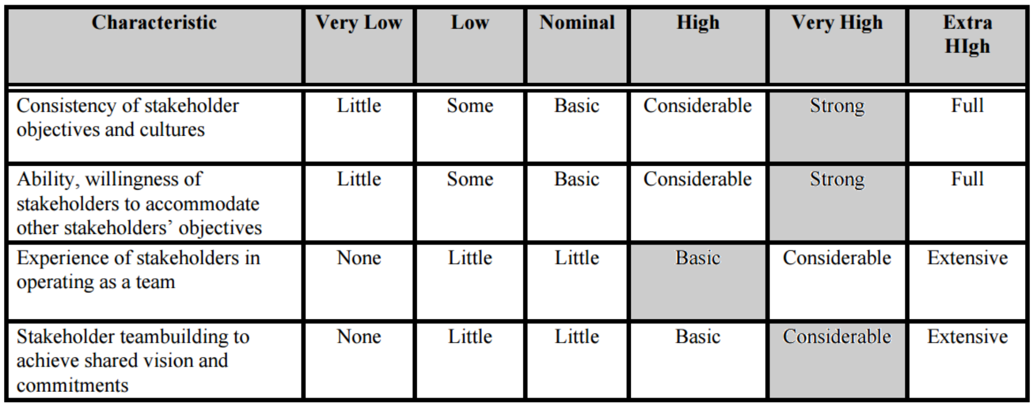
Process Maturity: Nominal We believe the CMMI level to be around "Defined", accounting for our relative lack of experience but proactive attitude.

The process was managed, planned and executed by skilled people in accordance to a policy, planned together with the stakeholders, and produced controlled output. This satisfies the CMMI level 2.



Characteristic	Very Low	Low	Nominal	High	Very High	Extra High
Risk Management Plan identifies all critical risk items, establishes milestones for resolving them by PDR.	None	[✓ Little]	Some	Generally	Mostly	Fully
Schedule, budget, and internal milestones through PDR compatible with Risk Management Plan	[✓ None]	Little	Some	Generally	Mostly	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

Figure 2: RESL checklist



Characteristic	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 teambuilding to achieve shared vision and commitments	None	Little	Little	Basic	[✓ Considerable]	Extensive

Figure 3: TEAM checklist

In addition to that, the project was also defined at the organizational level, since our project would be the Company's main interface to the world. Additionally, we did all we could to proactively identify and address possible obstacles in the process. This satisfies CMMI level 3.

2.2.3 Post-architecture Cost Drivers

Required Software Reliability: Nominal A service malfunction could cause inconvenience and non-trivial financial losses. This is because financial calculations are performed electronically, and a malfunction in the banking system could cause transaction losses. Additionally, Cars are locked and unlocked remotely, a lock service malfunction could lead to Cars being stolen.

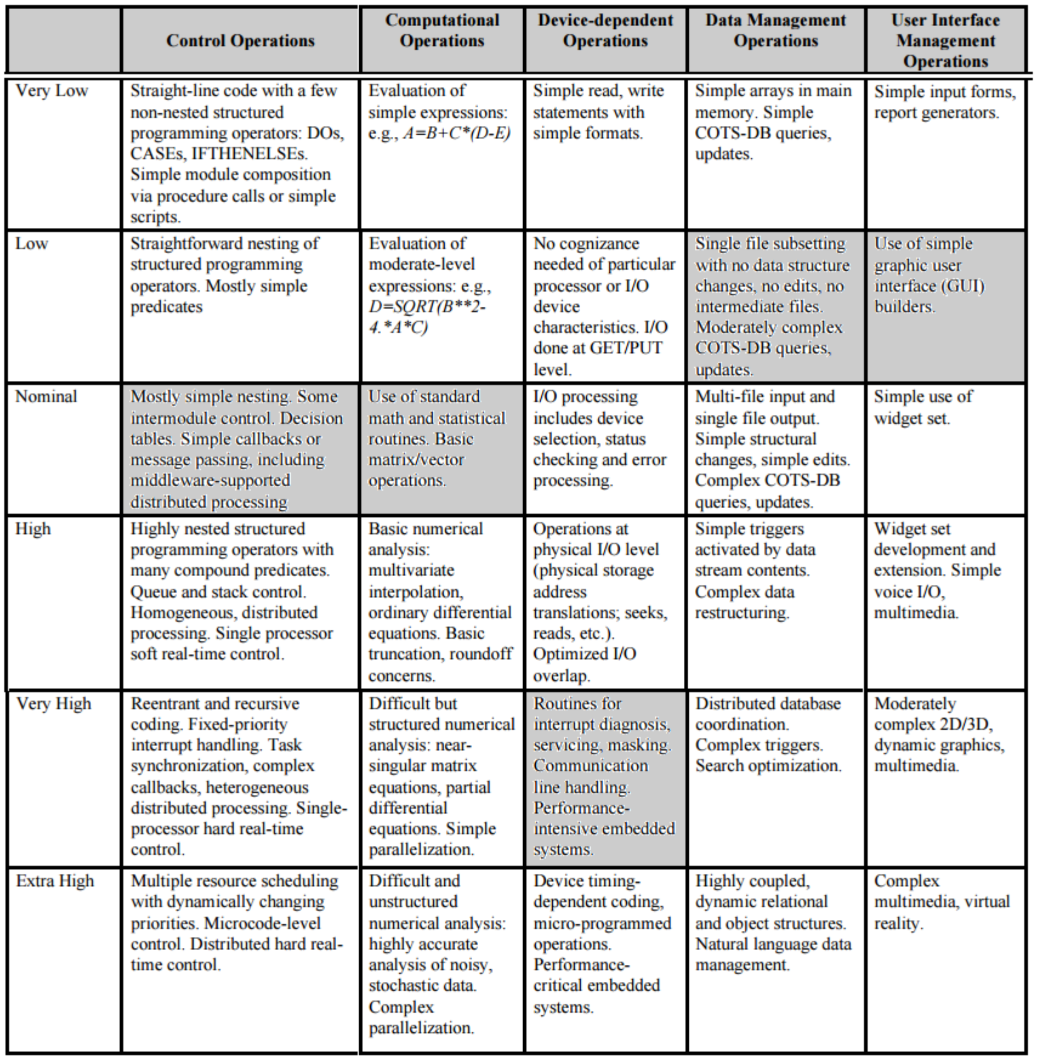
Data Base Size: Nominal The database size is standard for a medium-to-small-sized application. The database mainly stores data about Cars, Users, and Parking Areas. From an estimate of 12000 source lines of code, a nominal value is comprised between 120 KB and 1.2 MB, which is a reasonable estimate.

Product Complexity: Nominal The product's complexity is par for the course for a medium-to-small-sized application. COCOMO provides a table for calculating it, which is reported and marked on the next page. The most complex part of the application is that relying on communication with the remote devices.

Developed for Reusability: Nominal There was no specific reusability requirement, but we are working towards reusing components across the whole project whenever we do not need specialized components.

Documentation Match to Lifecycle Needs: Nominal Adequately-sized documentation will be provided for the various stages of the project.

Analyst Capability: Nominal We believe our skills to be around average, accounting for our relative lack of work experience. This project was designed during our course in Software Engineering 2, and we all passed Software Engineering I with good scores. We also cooperated extensively during the project.



	Control Operations	Computational Operations	Device-dependent Operations	Data Management Operations	User Interface Management Operations
Very Low	Straight-line code with a few non-nested structured programming operators: DOs, CASEs, IFTHENELSEs. Simple module composition via procedure calls or simple scripts.	Evaluation of simple expressions: e.g., *A=B+C*(D-E)*	Simple read, write statements with simple formats.	Simple arrays in main memory. Simple COTS-DB queries, updates.	Simple input forms, report generators.
Low	Straightforward nesting of structured programming operators. Mostly simple predicates	Evaluation of moderate-level expressions: e.g., *D=SQRT(B**2-4.*A*C)*	No cognizance needed of particular processor or I/O device characteristics. I/O done at GET/PUT level.	Single file subsetting with no data structure changes, no edits, no intermediate files. Moderately complex COTS-DB queries, updates.	Use of simple graphic user interface (GUI) builders.
Nominal	Mostly simple nesting. Some intermodule control. Decision tables. Simple callbacks or message passing, including middleware-supported distributed processing	Use of standard math and statistical routines. Basic matrix/vector operations.	I/O processing includes device selection, status checking and error processing.	Multi-file input and single file output. Simple structural changes, simple edits. Complex COTS-DB queries, updates.	Simple use of widget set.
High	Highly nested structured programming operators with many compound predicates. Queue and stack control. Homogeneous, distributed processing. Single processor soft real-time control.	Basic numerical analysis: multivariate interpolation, ordinary differential equations. Basic truncation, roundoff concerns.	Operations at physical I/O level (physical storage address translations; seeks, reads, etc.). Optimized I/O overlap.	Simple triggers activated by data stream contents. Complex data restructuring.	Widget set development and extension. Simple voice I/O, multimedia.
Very High	Reentrant and recursive coding. Fixed-priority interrupt handling. Task synchronization, complex callbacks, heterogeneous distributed processing. Single-processor hard real-time control.	Difficult but structured numerical analysis: near-singular matrix equations, partial differential equations. Simple parallelization.	Routines for interrupt diagnosis, servicing, masking. Communication line handling. Performance-intensive embedded systems.	Distributed database coordination. Complex triggers. Search optimization.	Moderately complex 2D/3D, dynamic graphics, multimedia.
Extra High	Multiple resource scheduling with dynamically changing priorities. Microcode-level control. Distributed hard real-time control.	Difficult and unstructured numerical analysis: highly accurate analysis of noisy, stochastic data. Complex parallelization.	Device timing-dependent coding, micro-programmed operations. Performance-critical embedded systems.	Highly coupled, dynamic relational and object structures. Natural language data management.	Complex multimedia, virtual reality.

Figure 4: CPLX checklist

Programmer Capability: Nominal We believe our skills to be around average, accounting for our relative lack of work experience. We have all completed several courses of programming with good grades, showing efficiency and thoroughness.

Personnel Continuity: Very High The three of us will keep working on the project from beginning to end.

Application Experience: Low We have little experience in developing complete applications, but we have been working on parts of them in our previous courses. We believe 6 months' experience to be a good estimate.

Platform Experience: Low We have little experience working with Spring, Tomcat and nginx. However, we have met several similar tools and platforms during our previous courses, so we believe 6 months' experience to be a reasonable estimate.

Language and Toolset Experience: Nominal We have a good understanding of Java, having used it across several courses. We are familiar with its built-in objects, main libraries, idioms, coding practices and with most Design Patterns. An estimate of 1 year's total experience is reasonable.

Time Constraint: Very High The Central Service will consume the vast majority of the execution time, being up and running all the time (barring crashes) and tracking all the Cars In Use. So will the communication services, since rapid communication between the Cars, the User Application and the Central Service is essential for good software performance.

Storage Constraint: Nominal No specific storage constraint was mentioned, and the amount of data created by the application is modest in size. We expect the application to use a storage percentage inferior to 50

Platform Volatility: Low No major changes to any of the platforms (JavaEE, Spring, Tomcat, nginx) are expected to happen during the project, since they are all stable, mature platforms. Minor changes are expected, mostly in the form of software upgrades and patches.

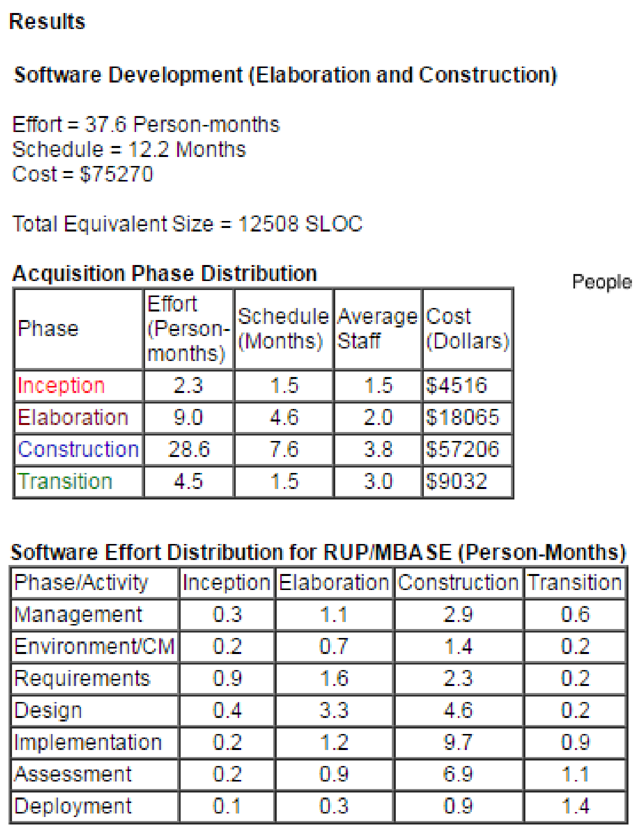
Use of Software Tools: High During development, several strong tools will be used to check on the lifecycle, such as maintaining a git repository. We will regularly update the repository with the latest source code, keeping the various design documents in the same place for easy access and accountability.

Multisite Development: Extra High Most of the work will be done from home, coordinating through IRC and telephone. This way, there will be no time wasted in looking for a place to work together.

Required Development Schedule: Nominal A rigid schedule was imposed on us for the first documents (RASD and DD), but we have more freedom for the project development. We will focus on concentrating our efforts in the early phases of development, in order to leave time for the unexpected.

2.2.4 Effort Calculation

By applying the algorithm (we used a free online tool, available at `http://csse.usc.edu/tools/COCOMOII.php`) based on our estimate of 236 FP, we obtained a result of 37.6 Person-Months, equivalent (considering a 160-hours work week) to 6016 Person-Hours.



Results

Software Development (Elaboration and Construction)

Effort = 37.6 Person-months
Schedule = 12.2 Months
Cost = $75270

Total Equivalent Size = 12508 SLOC

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	2.3	1.5	1.5	$4516
Elaboration	9.0	4.6	2.0	$18065
Construction	28.6	7.6	3.8	$57206
Transition	4.5	1.5	3.0	$9032

Staffing Profile

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

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.3	1.1	2.9	0.6
Environment/CM	0.2	0.7	1.4	0.2
Requirements	0.9	1.6	2.3	0.2
Design	0.4	3.3	4.6	0.2
Implementation	0.2	1.2	9.7	0.9
Assessment	0.2	0.9	6.9	1.1
Deployment	0.1	0.3	0.9	1.4

Figure 5: COCOMO tool results

2.2.5 Scheduling

COCOMO provides an estimate of the project time frame through the formula

$$Duration = 3.67 \cdot Effort^{SE}$$

where

$$SE = 0.28 + 0.2 \cdot (E - 0.91)$$

resulting in an estimate of 12.2 months.

2.2.6 Team Dimension

The dimension of the team can be easily calculated as $\frac{Effort}{Duration}$, which, in our case, yields a result of 3.1. Since there are 3 of us, the duration will likely be extended.

This is also due to the fact that the amount of staff needed in the different phases of development is not constant: the construction phase would optimally require 4 people working. This can be avoided by hiring an additional developer, or extending the deadline. In the latter case, a realistic adjusted duration would be equivalent to 14.2 months, to reflect the increased time needed for the software construction.

3 Task scheduling and resource allocation

For each of the required documents, we have split our tasks between ourselves. For the development of the application, we have separated a few large tasks and split them. These tasks are corresponding to the most significant and time-consuming work areas:

Central Services The main Server elaboration platform, including the business logic layer and External Services management.

Remote Devices The mobile app and the Car control device, the latter both in hardware and software.

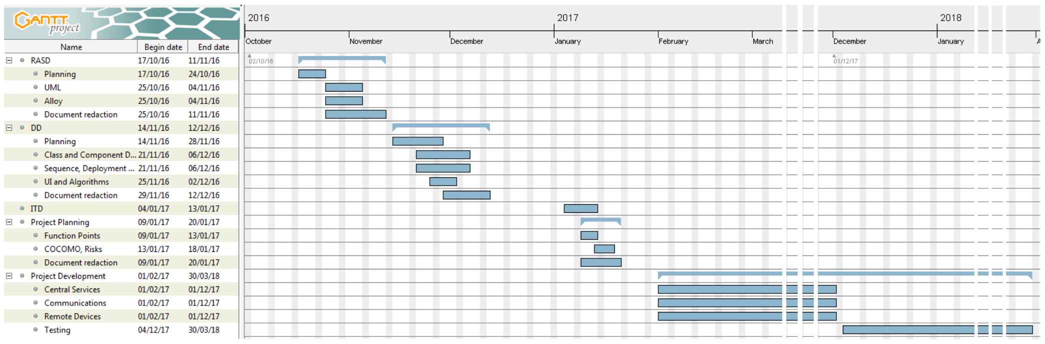
Communication The Persistence and Communication Layers, managing communication between the Central Services and the Remote Devices.

The following Gantt and resource diagrams show how these has been split. We are using the COCOMO-estimated development time, adjusted to reflect how our preliminary work was corresponding to the "Inception" development phase. Please not that the development part of the diagram has been cropped for space reasons.

The redaction of the Code Inspection document was not part of the PowerEnJoy project, therefore has not been included.

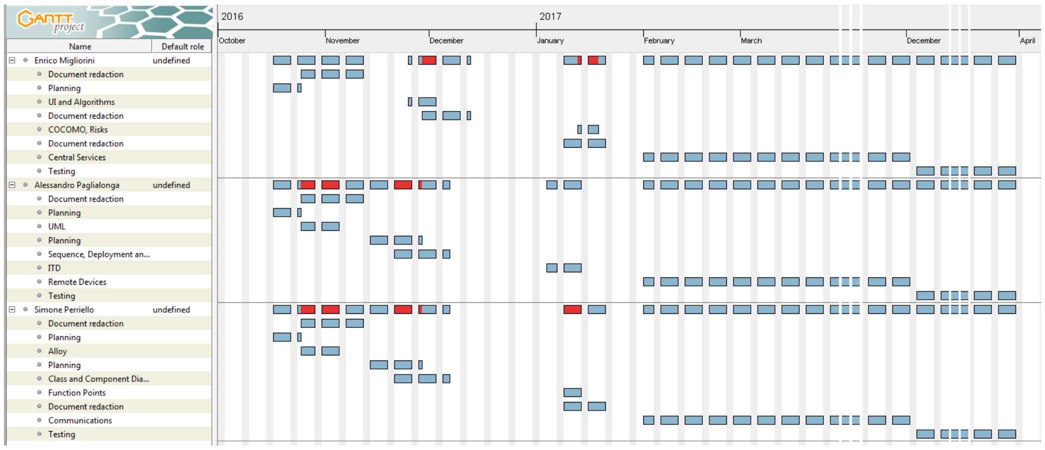
Major Task	**Start Date**	**Deadline**
Requirement Engineering (RASD)	16/10/16	13/11/16
Software Design (DD)	14/11/16	11/12/16
Integration Test (ITD)	04/07/17	15/01/17
Project Planning	09/01/17	22/01/17
Development	01/02/17	01/04/18

Table 9: Important project dates



Name	Begin date	End date
RASD	17/10/16	11/11/16
Planning	17/10/16	24/10/16
UML	25/10/16	04/11/16
Alloy	25/10/16	04/11/16
Document redaction	25/10/16	11/11/16
DD	14/11/16	12/12/16
Planning	14/11/16	28/11/16
Class and Component D...	21/11/16	06/12/16
Sequence, Deployment ...	21/11/16	06/12/16
UI and Algorithms	25/11/16	02/12/16
Document redaction	29/11/16	12/12/16
ITD	04/01/17	13/01/17
Project Planning	09/01/17	20/01/17
Function Points	09/01/17	13/01/17
COCOMO, Risks	13/01/17	18/01/17
Document redaction	09/01/17	20/01/17
Project Development	01/02/17	30/03/18
Central Services	01/02/17	01/12/17
Communications	01/02/17	01/12/17
Remote Devices	01/02/17	01/12/17
Testing	04/12/17	30/03/18

Figure 6: Gantt Diagram



Name	Default role
Enrico Migliorini	undefined
Document redaction	
Planning	
UI and Algorithms	
Document redaction	
COCOMO, Risks	
Document redaction	
Central Services	
Testing	
Alessandro Paglialonga	undefined
Document redaction	
Planning	
UML	
Planning	
Sequence, Deployment an...	
ITD	
Remote Devices	
Testing	
Simone Perriello	undefined
Document redaction	
Planning	
Alloy	
Planning	
Class and Component Dia...	
Function Points	
Document redaction	
Communications	
Testing	

Figure 7: Resource Allocation Diagram

4 Risk management

4.1 Possible risks

Code	**Name**	**Description**	**Chance**	**Impact**
R01	New competitors	A different company starts its own analogous service, so that our service is no longer needed.	Low	Serious
R02	Client's abandonment	Client withdraws interest and funding, stopping development before release.	Medium	Critical
R03	Lack of Budget	The allocated budget is insufficient to ensure development within schedule.	Medium	Serious
R04	Client's modifications	Client decides to change some significant detail of the application during development.	Medium	Marginal
R05	Customer's unforeseen needs	Customers need some feature that wasn't planned and included in the RASD.	High	Marginal
R06	Car-Mobile communication failure	Cars and Mobile applications can't communicate properly with the Central Server.	Medium	Serious
R07	Database failure	The Database is prone to errors, causing data to be unavailable or lost.	Low	Marginal
R08	Server failure	The Server is prone to problems, blocking the whole system.	Medium	Serious
R09	Developer's inability to work	Because of illness, personal problems or such, some of the developers can't work on the project any longer.	Medium	Critical

4.2 Preventive strategies

R01 An accurate market analysis has to be filled out by the Client before choosing to set the project up. We can't be held responsible for the needs of the market shifting.

R02 We will require, in the contract, a forward payment as well as a fee for early termination.

R03 We will need to provide a realistic estimation of budget. We will need to specify in the contract that a reduction in budget will result in a sub-par product or a longer development time.

R04 We will change the product according to the Client's desires. However, we'll calculate the workload needed for these changes and warn them of the consequent increase in development time and cost.

R05 This is a problem concerning Requirement Engineering. We will set up the system so as to make it easy to expand its functionalities to accomodate any unforeseen needs.

R06 We will have to invest time and effort towards having the hardware and software needed for communication working at top performance during development. We will document extensively that area in case modifications are needed.

R07 The DBMS is external. We recommend keeping a redundant backup database in order to be able to restore functionality immediately.

R08 We will build the Server systems for parallelism, in order to minimize the consequences of a crash. Also, we'll include in the documentation instructions on how to fix Server problems, so that the Company's IT Department could manage errors.

R09 We will consider replacements for the missing developer(s). However, even in the best case scenario, this will extend the development time because of communications overhead.

5 Work hours

5.1 Enrico Migliorini

Day	Work hours
14/01/17	1h
16/01/17	4h
17/01/17	5h
18/01/17	3h
19/01/17	2h
TOTAL	15h

5.2 Simone Perriello

Day	Work hours
14/01/17	3h
15/01/17	8h
22/01/17	4h
TOTAL	15h

5.3 Alessandro Paglialonga

He spent 1 hour on this document, mainly due to the planning of the tasks division.