

***“PowerEnjoy”***

*Project Management Document*

**Version 1.0** *(11/12/2016)*

Giorgio Marzorati (876546)

Aniel Rossi (877018)

Andrea Vaghi (877710)

INDEX OF CONTENTS

INTRODUCTION 2

Purpose & Scope 2

Reference Documents 4

PROJECT SIZE, COST AND EFFORT ESTIMATION 4

Size Estimation: Function Points 4

Internal Logic Files (ILFs) 4

External Interface Files (EIFs) 8

External Inputs (EIs) 10

External Inquiries (EQs) 12

External Outputs (EOs) 13

Overall size estimation 14

Cost and Effort Estimation: COCOMO II 14

Scale Drivers 14

Cost Drivers 16

Platform factors 18

Personnel factors 19

Project Factors 21

Effort equation: 23

Schedule and Resource Allocation 24

RISK MANAGEMENT 27

EFFORT SPENT 29

Giorgio Marzorati 29

Aniel Rossi 29

Andrea Vaghi 29

CHANGELOG 30

# **INTRODUCTION**

## Purpose & Scope

In this document, named Project Plan Document (PPD), we make a deep analysis on the complexity of the project in order to try to make the most real and feasible estimation of the costs and the effort involved in the entire developement. The main focus of this analysis is to estimate the required budget needed, the schedule of the activities and the allocation of the resources in order to cover all the duties.

In the first part of the document we use two techniques of size estimation (Function Points and COCOMO II) that help us to have an idea respectively on the final number of lines of code(SLOC) of the software and on the relation between the cost and the effort required to complete all the identified activities coherently with our planning.

In the second part we propose a schedule of the identified activities distributed along the period of time selected for the realization of the entire project, from the analysis of requirements to the final deployment and start up.

In the third part we assign each activity to one or more members of the team, based on the effort that we want to concentrate in them and the criticity that they represent respect to the entire project.

In the last part we make an evaluation of the risks in which our project could be involved. This analysis takes into account three main categories of risks:

* Project risks
* Technical risks
* Business risks

After the evaluation, we propose for each identified risk a mitigation strategy in order to handle the situation when the possibility of that risk becomes reality.

**Definitions, Acronyms, Abbreviations**

* FP: Function Points.
* ILF: Internal logic file
* ELF: External logic file.
* EI: External Input.
* EO: External Output.
* EQ: External Inquiries.
* DBMS: Database Management System.
* API: Application Programming Interface.
* ETA: Estimated Time of Arrival.
* UI: User Interface.
* GPS: Global Positioning System.
* RASD: Requirements and Analysis Specification Document
* DD: Design Document

## Reference Documents

• PowerEnjoy Requirement Analysis and Specification Document: RASD.pdf

• The Project Plan Example documents: Project Planning Example Document.pdf and Project Management Basics + Advanced.pdf

• The Function Points complexity evaluation tables.

• The COCOMO II Model Definition Manual (version 2.1, 1995 – 2000 Center for Software Engineering, USC).

# PROJECT SIZE, COST AND EFFORT ESTIMATION

In the next section we present a valuation of the expected size of the PowerEnjoy project (expressed in SLOC), then an estimate of the cost and the required effort, using the following two approaches:

* Function Points for the size estimation, based on each functionality that the business logic of the application has to offer and the corresponding lines of code to be written in order to correctly implement it;
* COCOMO II approach for the cost and the required effort, taking into account that we find ourselves into a Post-Architecture case of analysis

## Size estimation: function points

### Internal Logic Files (ILFs)

In this paragraph we describe the Internal Logic Files on which the PowerEnjoy data structure is based on, in order to be able to implement all the functionalities it offers. We focus in particular in the description of their complexity, to justify the amount of FPs we decided to apply to every single ILF.

ILF Rating (Low = 7 FPs, Avg = 10 FPs, High = 15 FPs):

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

Here is a brief description of the entities we use and the complexity assigned:

1) Client

Attributes:

* Driving License Id
* Email
* PIN
* Name
* Surname
* Card Circuit
* Card Expiration Date
* Card Number
* Card CVV

The Client file is composed of 1 single record with 9 data elements in total and we think we will reach a number of 10000 clients to be stored in the first year. By the way, since it can be represented with a single table, we assign a low complexity level.

2) Car

Attributes:

* License Plate
* Locked Status
* Availability
* Battery Level
* Last Latitude Position
* Last Longitude Position

As for the Client, the Car file can be represented with a single table, for a total amount of 6 data elements. Here we assign a low complexity level too.

3) Car Identification

Attributes:

* Identification Id
* Driving License Id (Client)
* License Plate (Car)
* Time of Identification

To correctly store a Car Identification, we need a dedicated entity with the time of the identification, the Client who performs it and the Car to be identified. The total amount of Data Elements is then 17 with 3 records element. Its complexity is low.

4) Car Reservation

Attributes:

* Reservation Id
* Driving License Id (Client)
* License Plate (Car)
* Time of Reservation
* Validity

A Car Reservation file is identified with a dedicated table that store the reservation time and its validity (expired or not), plus the information regarding the Client who performs it and the Car to be identified. So it requires 3 record elements and 18 data elements, and the complexity is low.

5) Ride

Attributes:

* Ride Id
* Driving License Id (Client)
* License Plate (Car)
* Start Latitude Position
* Start Longitude Position
* End Latitude Position
* End Longitude Position
* Start Time
* End Time
* Ride Charge
* Number of Passengers

The Ride file is quite more complex than the previous, because it still requires 3 record types (Ride, Car, Client) but the Data Elements required are higher (24). So we evaluated it with an average complexity.

6) Ride Charge

* Payment Id
* Ride Id (Ride)
* Amount
* Payment Evaluation (true/false)

and

7) Fee

* Fee Id
* Reservation Id (Reservation)
* Payment Evaluation (true/false)

The payments information we store about ride charged and fees for expired reservations are characterized by an average complexity. They in fact require 4 records element: Client, Car, Ride, plus Ride Charge for the first one and Fee for the second one. The number of Data Elements required is 20 for the Fee and 27 for the Ride Charge, so complexity is average for both.

8) Assistance Request

* Request Id
* License Plate (Car)
* Type
* Description
* Solved (true/false)

An Assistance Request is automatically created when sensors notice an anomaly in the car. The corresponding file is identified with 2 record elements, Assistance Request with the details and the Car. The complexity is low, as it needs 12 data elements.

9) Safe Area

* Safe Area Id
* Latitude Position
* Longitude Position
* Is Special (true/false)

Only the dedicated table uniquely identifies a Safe Area file, so it has a low complexity.

10) Power Grid

* Power Grid Id
* Safe Area Id
* Is Plugged (true/false)
* Is Busy (true/false)

To identify a Power Grid file, we use a dedicated table with the id, the availability and if it is actually plugged. We also need the information about the corresponding Safe Area, and so here there are 2 Record Element and 7 Data Elements. The complexity is low.

Overall FPs assignment:

|  |  |  |
| --- | --- | --- |
| **ILF** | **Complexity** | **FPs** |
| Client | Low | 7 |
| Car | Low | 7 |
| Car Identification | Low | 7 |
| Car Reservation | Low | 7 |
| Ride | Avg | 10 |
| Ride Charge | Avg | 10 |
| Fee | Avg | 10 |
| Assistance Request | Low | 7 |
| Safe Area | Low | 7 |
| Fee | Low | 7 |
| *Total* | | 79 |

### External Interface Files (EIFs)

In this section we explain the interaction between PowerEnjoy and some sets of data provided by other independent applications/providers. In particular our system relies on two main external services, the *Mapping Service* and the *Payment Service Provider*. The information retrieved through the interaction between the system and these components sometimes need a certain amount of analysis and processing in order to be useful in our system. There may be situations in which the complexity of some of those interactions needs to be considered in a proper way, taking into account the dimension of the data exchanged and the amount of business logic required to process it. The list of the interactions is the following:

EIF Rating (Low = 5 FPs, Avg = 7 FPs, High = 10 FPs):

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

**Mapping Service:**

* City Map Retrieval:

This information is requested for the displaying of the city map on the user’s mobile application. The operation of retrieving is quite simple, because it requires just the user position or an address, but the amount of data returned is high and needs some logic to be forwarded to the client. Thus, we decided to assign a high complexity value

* Address - Position Conversion Result

The Mapping Service is able to convert an address provided in String format to a pair of coordinates that will be used to update a single table on the database for each request, so the complexity of this EIF is low

* Money Saving Mode Data:

Money saving mode data is a collection of information that the user gets when activates that mode (ETA, Distance to the destination, Information about the path and so on). We need some logic operation in order to provide to the client the information it requires in the right way. So we decided to assign a medium grade of complexity.

* Distance Retrieval:

The distance between two pairs of coordinates is a simple data expressed with a double precision floating point value, computed by the Mapping Service. As inputs, that external component needs only the coordinates representing the start point and the end point. Since we only need to provide this kind of information (that we mainly store in the database) and we don’t have to apply any logic to the returning data (maybe just some formatting operation), we classify this operation with a low complexity

**Payment Service Provider:**

* Transaction Acknowledgement:

The only information we receive from the Payment Service Provider is an acknowledgment for the outcome of a user payment for a ride or a fee for an expired reservation. Since we store the payments in a single ILF, all we have to do with the data incoming from the Payment Service Provider is an update on a single field of a table on the database, we assume that the complexity of this operation can be evaluated as low

Overall FPs assignment:

|  |  |  |
| --- | --- | --- |
| **EIF** | **Complexity** | **FPs** |
| City Map Retrieval | High | 10 |
| Address/Position Convertion | Low | 5 |
| Money Saving Mode Data | Avg | 7 |
| Distance Retrieval | Low | 5 |
| Transaction Ack | Low | 5 |
| *Total* | | 32 |

### External Inputs (EIs)

In our PowerEnjoy system, the back-end section interacts with different categories of client, providing a set of functionalities in response of several types of inputs.

In this section we focus on the complexity of each EI (grouped by clients), explaining our considerations in every case of analysis.

EI Rating (Low = 3 FPs, Avg = 4 FPs, High = 6 FPs):

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Data Elements** | | |
| **File Type Referenced** | 1-4 | 5-15 | 15+ |
| 2- | Low | Low | Avg |
| 2 | Low | Avg | High |
| 2+ | Avg | High | High |

**End User:**

* Registration:

The user registration involves the data processing by the Registration Manager, some, database checks, a record insert and the Notification Manager effort for sending the PIN. As we can notice, it is not a straightforward operation, so it can be considered with an average complexity.

* Login/Logout:

The user Login & Logout operations are quite simple, as they affect just the Login Manager and include simple database operations (a check and a basic update). The overall complexity is low (3 FPs for each operation).

* Lost Password Retrieval:

If a user loses its password, there are different steps in order to deliver a new one: database operation to delete and update fields, generation of a new one by the Account Manager and the Notification Manager to communicate it to the user. This complexity is evaluated as average.

* Car Reservation:

This operation involves the Reservation Manager to perform a series of tasks, as it has to create a new Reservation entity based on the information provided by the user (driving license, car license plate) and instantiate a timer that determines the validity of a reservation for 60 minutes. It also has to update the availability of a car in the database. We assign an average level of complexity.

* Car Identification:

Identification of a car by a user is a complex operation. In fact, to achieve successfully this task, different components (and the respective managed entity on the database) have to be involved: the Identification Manager has to send a message to the Car Manager that forwards it to the right car, unlocking it physically. The Identification Manager has also to communicate to the Reservation Manager, in order to be sure that a car is actually available and to end a possible reservation made by the same user who performs the identification. Due to these considerations, we assign a high level of complexity.

* Money Saving Mode Enabling:

Money Saving Mode data is a collection of information that the user gets when activates that mode (ETA, Distance to the destination, Information about the path and so on). Although this data are computed by the mapping service, we need several logic operations in order to provide to the external service the information it requires (two pairs of coordinates, one for the user actual position and one for the Special Parking Area). Specifically, the choice of the second one is based on the final destination of the user, the availability of Special Parking Areas and the distribution of cars, so to suggest the right place to park we need several database queries, distance computation and comparisons. Taking into account these considerations, we decided to assign a high level of complexity.

**Car:**

* Generic Sensor Data Transmission

The data sensor transmission involves the Car Manager that receives and processes them to check if they respect some parameter (if the car is damaged or has some kind of issue. If not, it has to interact with the Assistance Manager for creating an automatic reporting that has to be stored in the database. The complexity is average.

* Ride Start Data Transmission

When a ride starts, the car transmits different data to the Car Manager (start position, battery level, number of passengers) which has to interact with the Ride Manager for the creation of a Ride entity. It is not a very complex operation, but it requires some effort due to checks and data collecting from different entities, so we assign an average level of complexity.

* Ride End Data Transmission

The end of a ride implies a high effort by various components the same way as for the ride start, but with the addiction of all the processes to determine the charge to be paid, including discount and overcharge calculation and the direct interaction with the Payment Manager. For this reason, we evaluated this complexity as high

**Assistance Team:**

* Telephone Assistance Requests Inserting

The assistance team’s task that include a direct interaction with the Central Server is the one regarding the insert in the system of the requests they receive via telephone. It is a simple insert or update operation, translated by the server in a database statement, so we rate it as a low complexity one.

Overall FPs assignment:

|  |  |  |
| --- | --- | --- |
| **EI** | **Complexity** | **FPs** |
| Registration | Avg | 4 |
| Login/Logout | Low | 2x3 |
| Lost Password Retrivial | Avg | 4 |
| Car Reservation | Avg | 4 |
| Car Identification | High | 6 |
| Money Saving Mode Enabling | High | 6 |
| Generic Sensor Data Transmission | Avg | 4 |
| Ride Start Data Transmission | Avg | 4 |
| Ride End Data Transmission | High | 6 |
| Teleph. Assistance Req. Inserting | Low | 2x3 |
| *Total* | | 50 |

### External Inquiries (EQs)

In this paragraph we analyse the External Inquiries (EQs) influencing our system, that are basically data retrieval request by an user, without a big amount of business logic being involved in the process. PowerEnjoy supports some of these kind of operations, everyone characterized by a low complexity level since here is not involved big amount of data and the effort for retrieval is quite low (except for the car retrieval, which needs the comparison between the user position and the car position by the Mapping Service):

* A user can request the available car in an area around its position or an inserted address
* A user can request the history of its past reservations
* A user can request the history of its past rides
* A user can request the history of its past payments (fees and ride charges)
* The Assistance Team can request assistance requests with several type of filters (by cars, users, zones, times etc)

Overall FPs assignment:

|  |  |  |
| --- | --- | --- |
| **EIF** | **Complexity** | **FPs** |
| Available Cars Request | Avg | 4 |
| Past Reservation Request | Low | 3 |
| Past Reservation Ride | Avg | 3 |
| Past Reservation Payments | Avg | 3 |
| Assistance Requests | High | 3 |
| *Total* | | 16 |

### External Outputs (EOs)

In the last part of the Function Point analysis we focus on the External Outputs, that are situation in which PowerEnjoy needs to communicate with the external environment, sending various types of information. This communications are:

* PIN notification to the user:

A simple operation via E-Mail that involves just the Notification Manager and the Database Manager, to set up a right PIN creation. Complexity: low.

* Fee notification to a user:

It’s a simple E-Mail message sent to the user by the Notification Manager when a Reservation expires. Complexity: low.

* Car Unlocking:

It’s a simple outcome sent to the Car System by the Car Manager, to perform the unlocking. Complexity: low.

* Fee and Ride Charge Payment Details:

The Payment Manager sends this info to the Payment Service Provider. The system needs to collect some data in order to achieve this task (user’s billing details, amount to be paid), retrievable from few tables in the database. Complexity: average for both.

* Assistance Request notification

The Assistance Manager has to collects sensors data passed by the Car Manager, and information about the car (license plate, position), attaching the type of problem (interpreting sensor’s data) and pass them to the Assistance Team System. Complexity: average.

Overall FPs assignment:

|  |  |  |
| --- | --- | --- |
| **EIF** | **Complexity** | **FPs** |
| PIN Notification | Low | 4 |
| Fee Notification | Low | 4 |
| Car Unlocking | Low | 4 |
| Fee/Ride Charge Payment Details | Avg | 2x5 |
| Assistance Requests | Avg | 5 |
| *Total* | | 27 |

### Overall size estimation

|  |  |
| --- | --- |
| **Function Type** | **FPs** |
| Internal Logic Files | 79 |
| External Interface Files | 35 |
| External Input | 50 |
| External Inquiries | 16 |
| External Output | 27 |
| *Total* | 207 |

We consider JEE as our platform for the development for the entire business logic. The client’s mobile application is not included in this calculation, since we designed our system to contain the business logic in the back-end side.

Estimated lines of code:

**Lower bound - SLOC = 207 \* 46 = 9522**

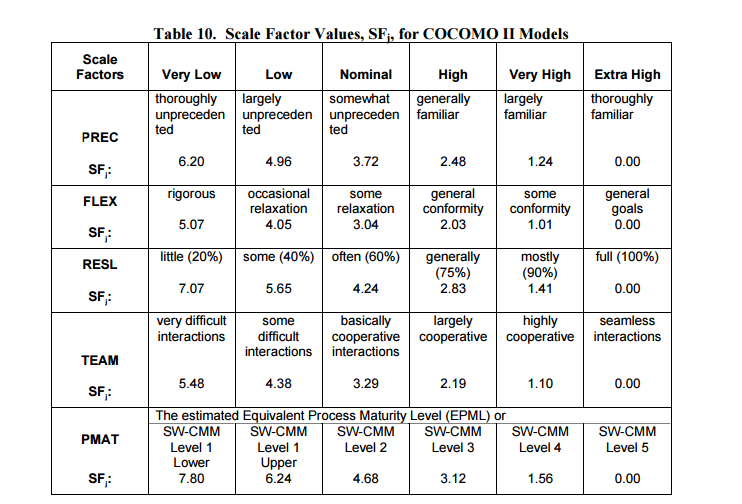
**Upper bound - SLOC = 207 \* 67 = 13869**

## COST AND EFFORT ESTIMATION: COCOMO II

In this section we use the second estimation method mentioned in the introduction, COCOMO II. The purpose is to make an analysis of the costs and the effort needed in order to be able to build the project in terms of time and money.

### Scale Drivers

The first thing we did was evaluating some scale factors, making a sort of highlighting of some of the main properties of the project and the team. For this purpose we refer to the following official COCOMO II table (next page):



A brief description for each scale driver:

• Precedentedness: it reflects the previous experience of our team with the development of large scale projects. In our case the value is low, our team made together only one another large scale project except this.

• Development flexibility: it reflects the degree of flexibility in the development process with respect to the external specification and requirements. Since we don’t have any particular constraint (especially technical) specified in the specifications the flexibility is very high.

• Risk resolution: reflects the level of awareness and reactiveness with respect to risks. The risk analysis we performed is quite extensive, evaluating both internal and external risks, dividing them also in categories. The value is set to high.

• Team cohesion: it’s an indicator of how well the team members know each other and work together in a cooperative way. Here the value is high due to the fact that we know each other very well and we did another project together as a team before .

• Process maturity: due to the fact that we work on a well defined development process this value is set to nominal.

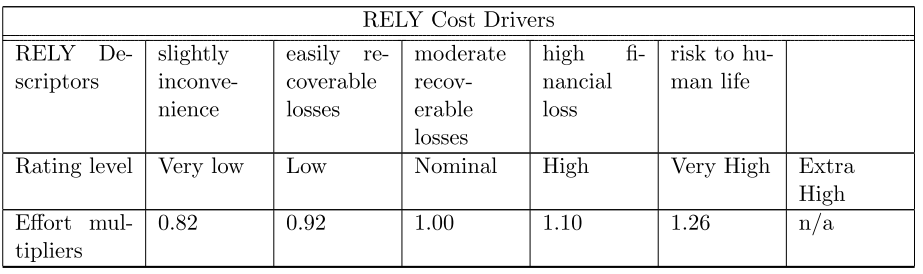
The results of our evaluation is the following (next page):

|  |  |  |
| --- | --- | --- |
| **Scale Driver** | **Factor** | **Values** |
| Precedentedness | Low | 4.96 |
| Development Flexibility | Very High | 1.01 |
| Risk Resolution | High | 2.83 |
| Team Cohesion | High | 2.19 |
| Process Maturity | Nominal | 4.68 |
| *Total* | | 15.67 |

### Cost Drivers

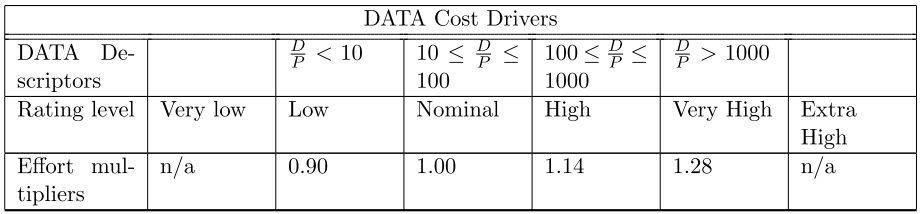
*Reliability*:

A possible shutdown of the server in a particular amount of time leads only to the loss of the payments of the rides that are happening in that moment and the impossibility to reserve or identify a car. In the case of missed payments we can easily recover the rides that weren’t paid from the database. In our case the value is set to Low.



*Database Size*:

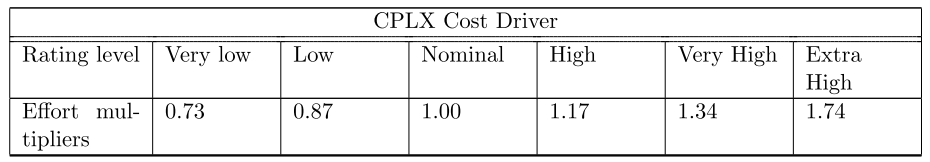
This measure considers the effective size of our database. Our estimation considering the tables and fields we have is to reach a 3GB database. Since it is distributed over 9.000-14.000 SLOC, the ratio D/P (measured as testing DB bytes/program SLOC) is between 209 and 314, resulting in the DATA cost driver being High.



*Product Complexity*:

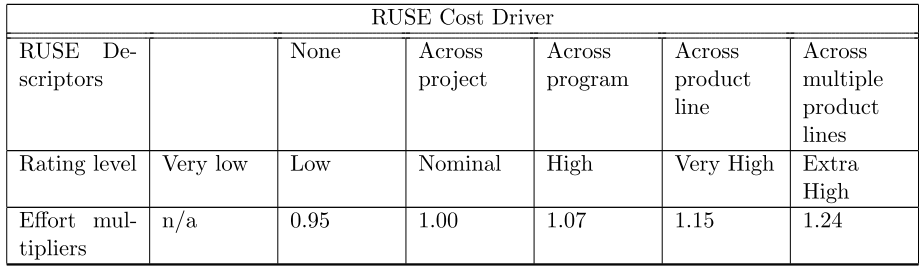
Complexity is divided into five areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations.

In our case the functions will be mostly functions with normal complexity of calculus and construction and they will be simple to use and understand. The value is set to Nominal.



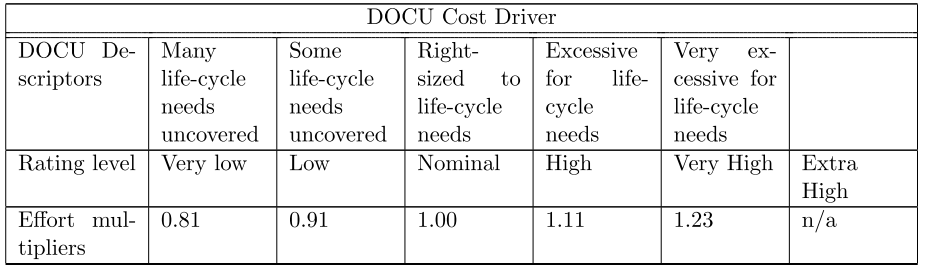
*Required Reusability*:

This cost driver accounts for the additional effort needed to construct components intended for reuse on current or future projects. In our case, the reusability requirements are limited in scope to the project itself, so the RUSE cost driver is set to Nominal.



*Documentation match to life-cycle needs*:

This parameter describes the relationship between the documentation and the application requirements. The rating scale goes from Very Low (many life-cycle needs uncovered) to Very High (excessive for life-cycle needs). In our case the value is Nominal, due to the fact that in the documentation are covered the main life-cycle needs, so the document descriptors can be considered right-sized to life-cycle needs.

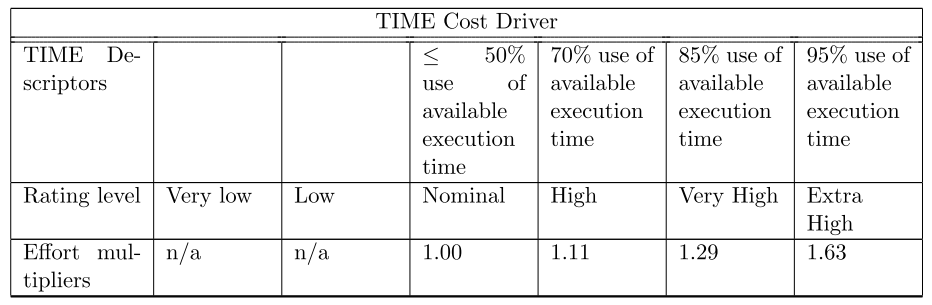


### Platform factors

These types of factors refer to the complexity of the hardware talking about the machine level.

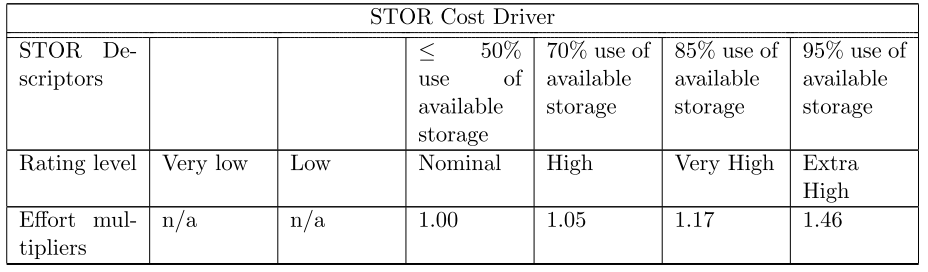
*Execution Time Constraint*

This is the measure of the execution time constraint imposed upon a software system. Since we do not have many accesses in the same moment, we are not going to handle too many parallel operations and these are going to be simple one. For these reasons, we choose the value High of the table corresponding to the rate 1.11. The rating express the percentage of the available execution time expected to be used by the system or subsystem consuming the execution time resource.



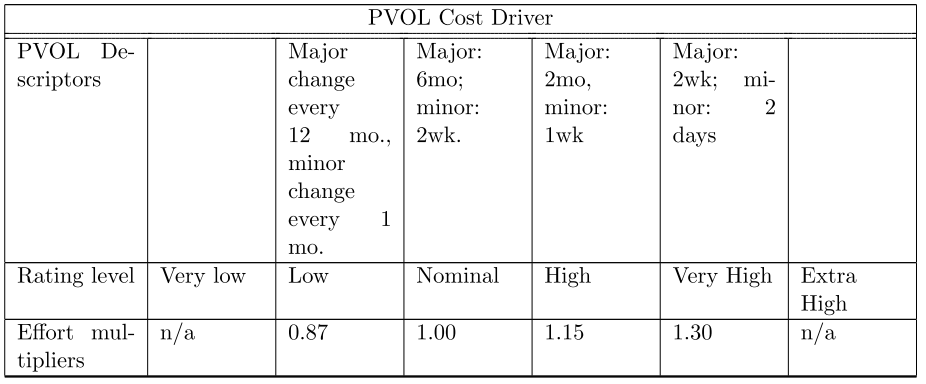
*Main storage constraint*

This rating represents the degree of the main storage constraint imposed on a software system or subsystem. Because of the low cost and the high capacity of disks on the market, we consider this parameter as Nominal.



*Platform volatility*

In this section with platform we are going to mean the complex of hardware and software (form OS to DBMS) that the software has to call to perform its tasks. Since we would like to be aligned with all the software products on which we rely on (car software, payment method, OS client), but there are in a small number of them we can imagine to have not so many changes during the time, for that reasons we get a Nominal parameter (1.00).

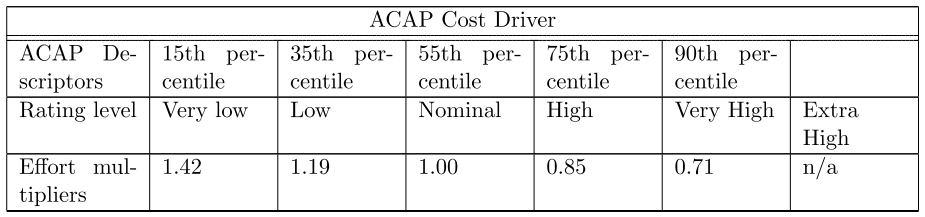


### Personnel factors

People factors have the strongest influence in determining the amount of effort required to develop a software product and they rate the development team’s capability and experience.

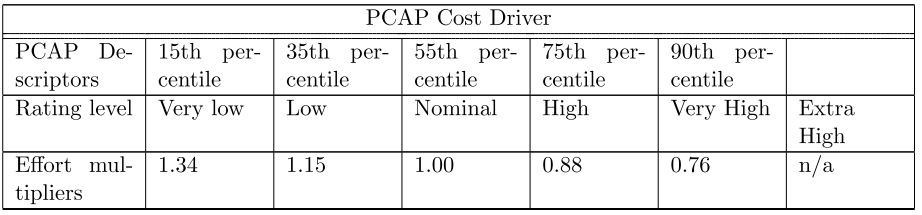
*Analyst capability*

This value represents the design ability and efficiency, the ability to communicate and cooperate of the analyst, personnel who work on requirements and high-level design and detailed design. Since we spent a lot of time analysing the requirements deeply and building a solid system during the requirement analysis and the design phase we could think that we are going to have less problem in the future phase for the development so we choose the High value with rate 0.85.



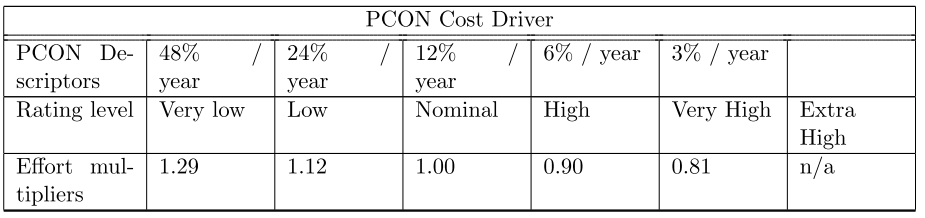
*Programmer capability*

Since with this value we represent the ability, the efficiency and the thoroughness of our team of programmers we set an High value (0.88) because we consider us an efficient team, since we had some previous experiences of this kind together.



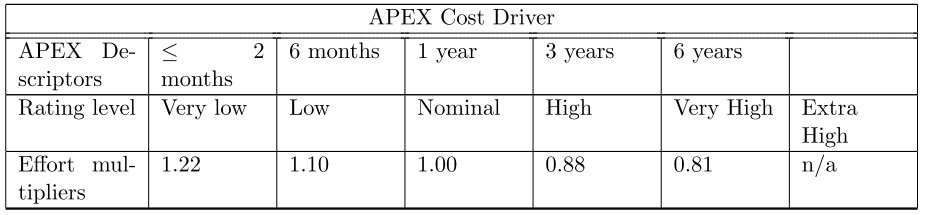
*Personnel continuity*

Considering our period of work and development of the project, we think that we are going to be present during almost every meeting, so we choose the High value.



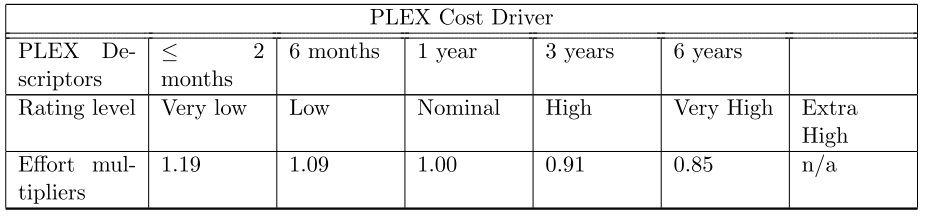
*Application experience*

This rating is dependent on the level of applications experience of the project team developing the software system or subsystem in term of year of experience. We choose a Nominal value of 1.00 because we developed together another project before this one.



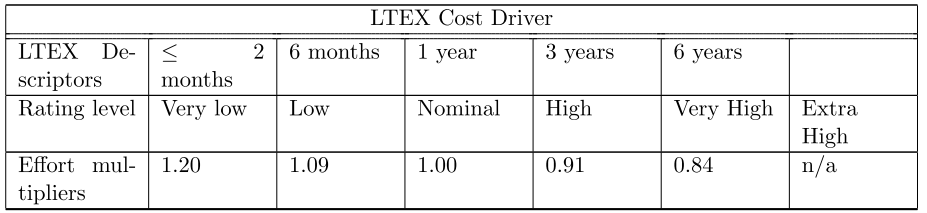
*Platform experience*

With this value, we consider the platform (graphic user interface, database, etc.) and knowledge experience. Because of our past project involving databases, graphic user interfaces, we would choose a Nominal value, but because of our non-deep knowledge of these platforms we prefer setting the value to Low(1.09).



*Language and tool experience*

This rating depends on the level of knowledge and experience of the programming language and the software tool used, including the ones used in the requirements and design phases, configuration analysis, document extraction and so on. Because of our average knowledge, we set the value to Nominal.

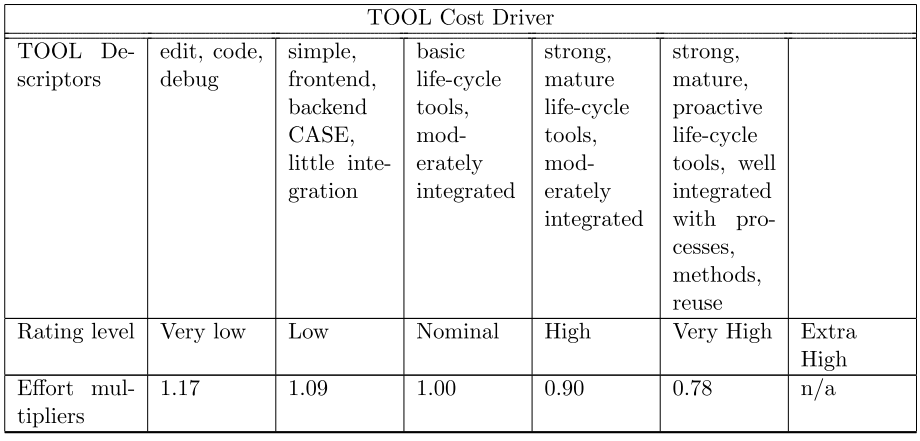


### Project Factors

Account for influences from the use of modern software tools, compression of the project schedule or location of development team.

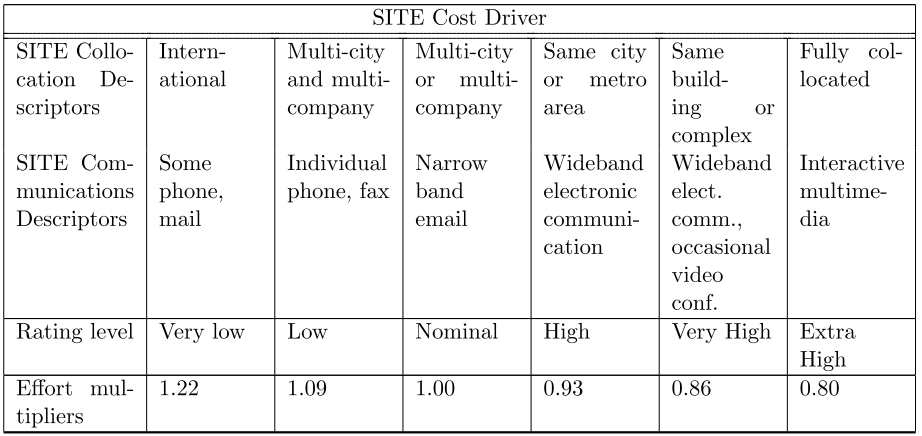
*Use of software tools*

It reflects the tool capability, maturity and integration. Because we are going to use solid tools and we are not going to use open source ones, which could create some problems, we choose the High value.



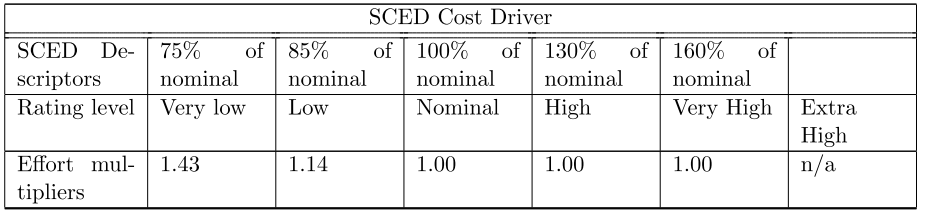
*Multisite development*

Since we program to build the great part of the system during our meetings we consider the Very High parameter.



*Required development schedule*

This rating measures the constraint imposed on the project team developing the software in terms of percentage of schedule stretch-out or acceleration with respect to the nominal schedule. Because of our hard work during requirements and design phases we think that the right value is the High one.



### Effort equation:

Effort = A \* EAF \* KSLOC^E

Our results:

E=1.07

Size = KSLOC -> Lower bound = 9.52 Upper bound = 13.87

0.66

Lower bound effort = 2.94 \* 11.15 \* 0.66 ≈ **22 PM**

Upper bound effort = 2.94 \* 16.67 \* 0.66 ≈ **32 PM**

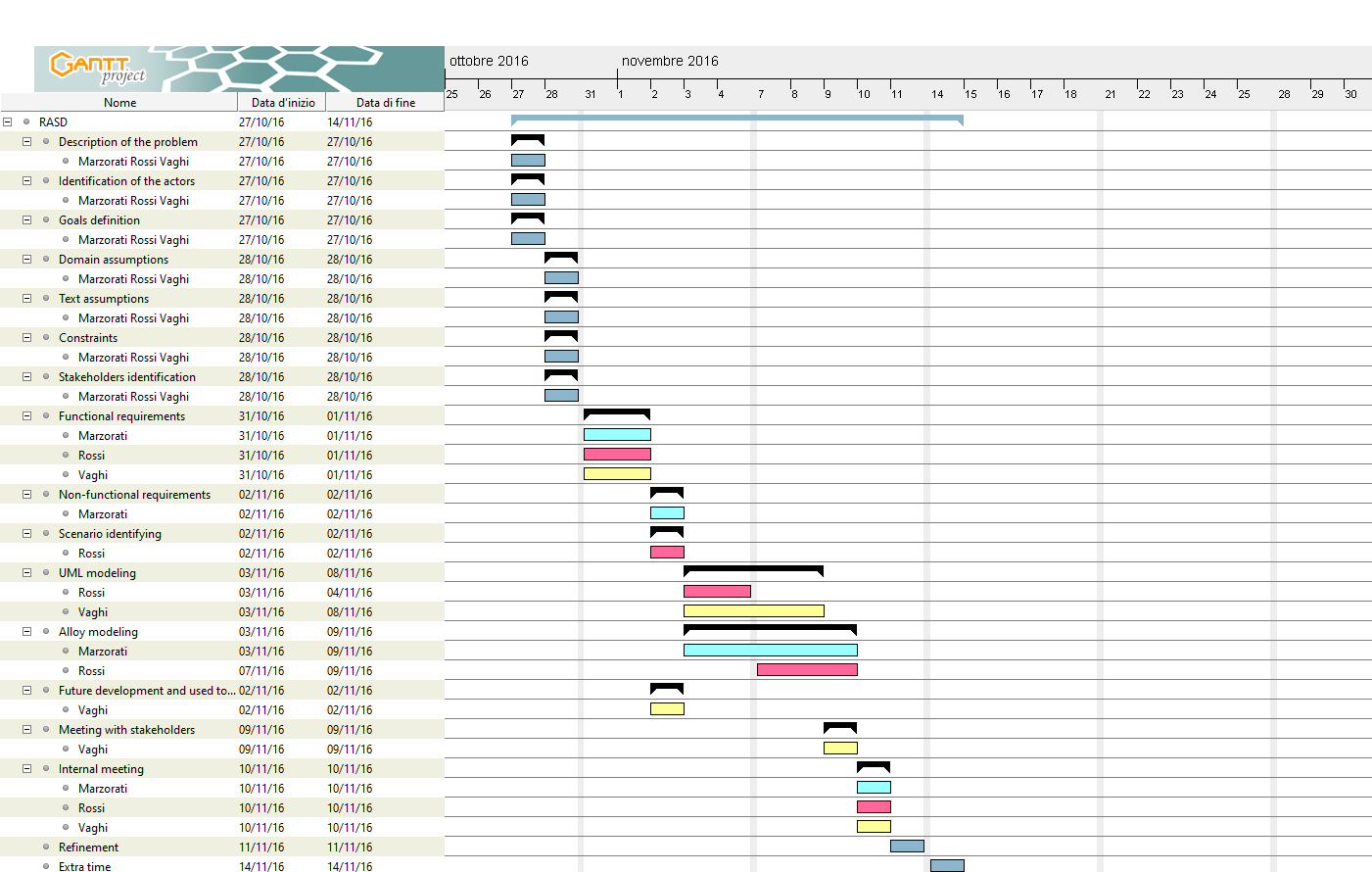
## Schedule and Resource Allocation

In this chapter we provide a general time scheduling of all the activities involved in this project from the very beginning (requirement analysis) to the end (start-up).

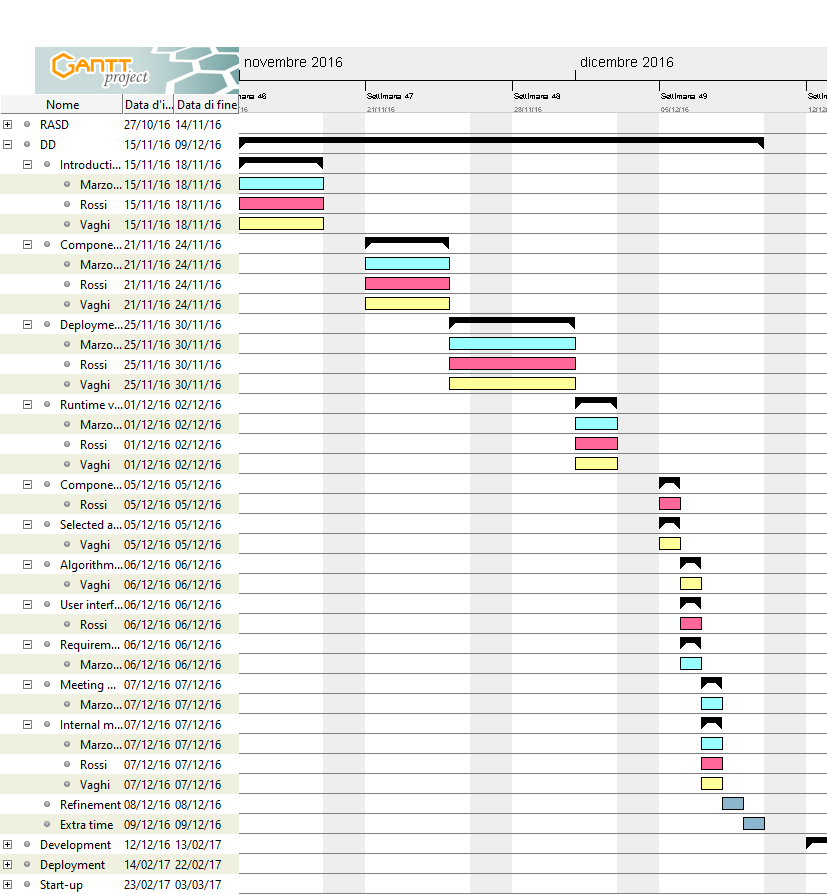
In the same diagram we also specify the percentage of work that each member will spend on each of these tasks.

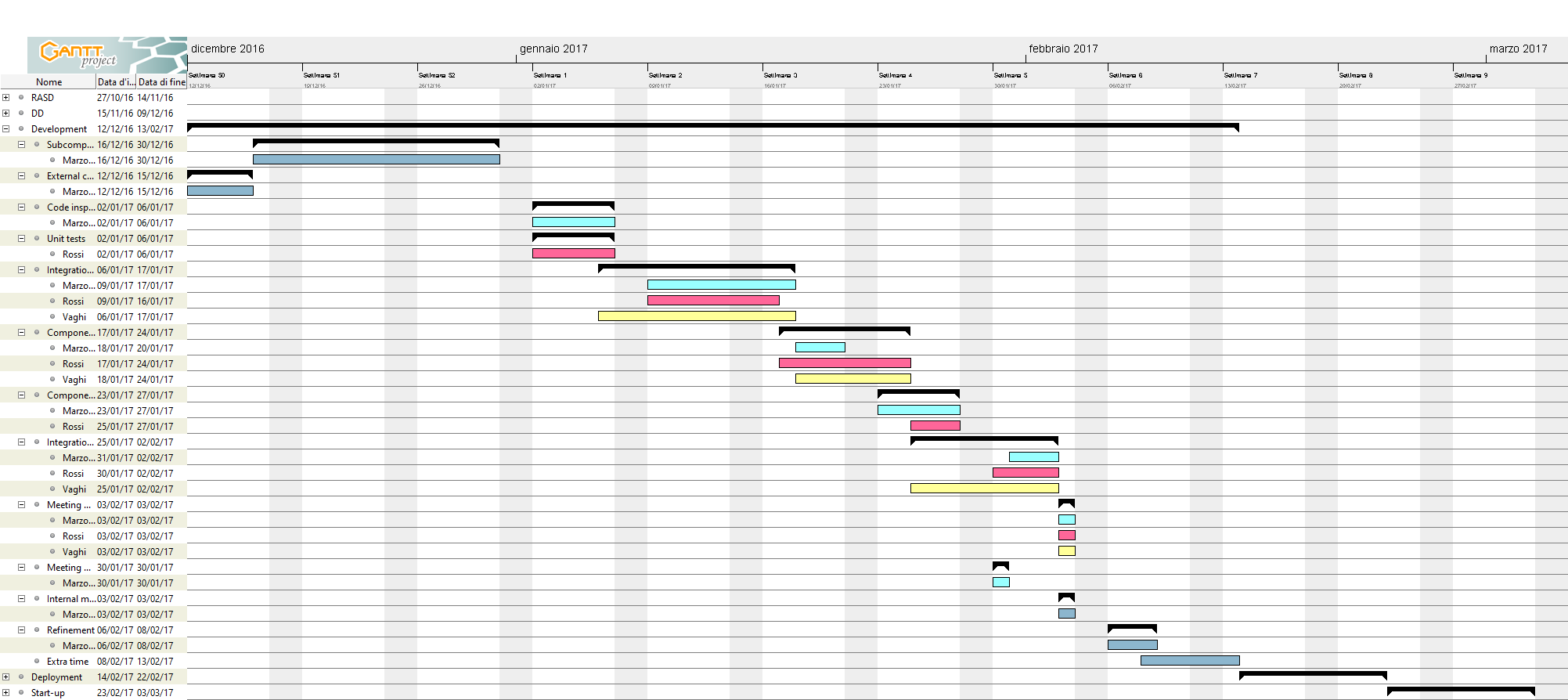
The amount of time considered is from October to March.

RASD:

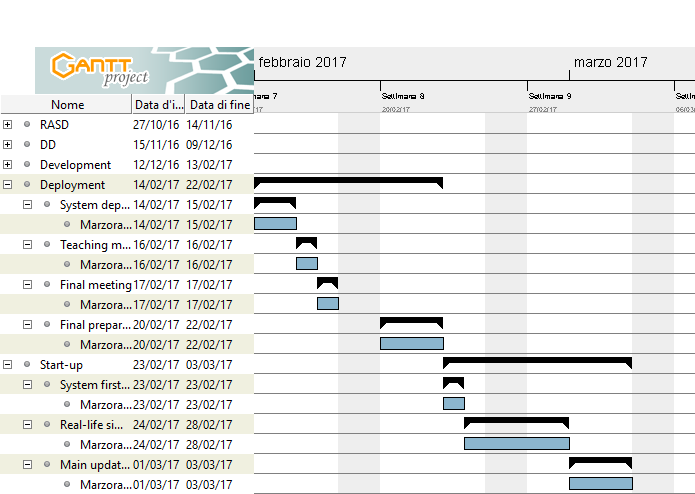


DD:



Development:

Deployment to the end:



# RISK MANAGEMENT

In this chapter we analyse some of the risks that we may face during the project development, dividing them in 3 categories: project risks, technical risks and business risks.

1) Risk Description:

|  |  |  |  |
| --- | --- | --- | --- |
| **Code** | **Risk** | **Probability** | **Effect** |
| PR1 | A key member of the team quits or is ill in a critical point of the project development | Average | Serious |
| PR2 | Some parts of the project require more time than expected (underestimating schedules) | Average | Serious |
| PR3 | The external car software component doesn’t work as expected or is not ready in time | Low | Catastrophic |
| PR4 | Loss of entire or part of the source code | Low | Catastrophic |
| TR1 | External commercial components change their configurations/protocols/API | Low | Catastrophic |
| TR2 | A new paymenth method starts getting used widely | Average | Serious |
| BR1 | Change in available car parking zone | Average | Serious |
| BR2 | The project don’t satisfy the expectation of the city amministration | Average | Catastrophic |
| BR3 | Budget overestimation | High | Serious |
| BR4 | Budget underestimation | Average | Catastrophic |
| BR5 | Missing agreement with the mobile data vendor | Average | Catastrophic |

2) Mitigation:

|  |  |
| --- | --- |
| **Code** | **Mitigation strategy** |
| PR1 | Split duties and responsibilities among the highest number of people. In the worst case we can consider a first beta release with less functionalities than the defined ones during the requirement analysis phase |
| PR2 | Schedule an extra time slots before the major activities or releases |
| PR3 | A bunch of meetings with the external sw company to define deadlines |
| PR4 | Use of a backup system to store versions of data |
| TR1 | Try to build components that are independent among them (modularity of the system) with a particular focus on interfaces implementation |
| TR2 | Keep sistematically under control the new methods of payment on the market |
| BR1 | Exploit the time between the proposal and the confirmation of the measure to update our system |
| BR2 | Make the administration part of the development process (meetings and presentations on RASD and DD) |
| BR3 | Propose an alternative way of using the residual budget (application support, future releases, new functionalities) |
| BR4 | First release with less functionalities than the defined ones during the requirement analysis phase |
| BR5 | Exploit the economy of scale in the market of mobile data providers |

# EFFORT SPENT

## Giorgio Marzorati

29/11/2016 - 3h

04/12/2016 - 3h

05/12/2016 - 2h

08/12/2016 - 3h

09/12/2016 - 4h

10/12/2016 - 3h

11/12/2016 - 2h

## Aniel Rossi

29/11/2016 - 3h

04/12/2016 - 3h

05/12/2016 - 2h

08/12/2016 - 3h

09/12/2016 - 5h

10/12/2016 - 3h

11/12/2016 - 1h

## Andrea Vaghi

29/11/2016 - 3h

01/12/2016 - 2h

02/12/2016 - 3h

04/12/2016 - 1h

05/12/2016 - 2h

06/12/2016 - 1.30h

07/12/2016 - 1h

08/12/2016 - 3h

09/12/2016 - 1h

10/12/2016 - 5h

11/12/2016 - 3h

# CHANGELOG

V1.0 - First release