

PROJECT for

SOFTWARE ENGINEERING 2

**Project Management v1**

Politecnico di Milano

A.A. 2016-2017

Prof.ssa Elisabetta Di Nitto

Students:

Diego Gaboardi

Giorgio Giardini

Riccardo Giol

INDEX

* 1 Introduction
  + 1.1 Purpose and Scope
  + 1.2 Definition, Acronyms
    - 1.2.1 Definitions
    - 1.2.2 Acronyms
  + 1.3 Reference documents
* 2 Project size, cost and effort estimation
  + 2.1 Introduction
  + 2.2 Size estimation: functions points
    - 2.2.1 Internal Logic Files (ILFs)
    - 2.2.2 External Interface Files (EIFs)
    - 2.2.3 External Inputs (EIs)
    - 2.2.4 External Inquiries (EQs)
    - 2.2.5 External Outputs (EOs)
    - 2.2.6 Overall estimation
  + 2.3 Cost and effort estimation: COCOMO II
    - 2.3.1 Scale Drivers
    - 2.3.2 Cost Drivers
    - 2.3.3 Effort equation
    - 2.3.4 Schedule estimation
* 3 Schedule
* 4 Resource Allocation
* 5 Risk management

1 INTRODUCTION

* 1. PURPOSE AND SCOPE
  2. DEFINITIONS AND ACRONYMS
  3. REFERENCE DOCUMENTS

CII\_modelman2000.0.pdf

1. SIZE, COST AND EFFORT ESTIMATION
   1. INTRODUCTION

In this section we are going to estimate the following properties about our project: the expected size, the cost and the required effort.

So we are going to use the Function Points approach in order to obtain a good estimation of the size of our project in term of hypothetic number of lines of code.

Then for the cost and effort estimation we will use the COCOMO approach. In this way starting from the number of lines of code we will estimate the number of person hours necessary for developing the whole project.

* 1. SIZE ESTIMATION: FUNCTION POINTS

The Function Points approach allows us to obtain a good estimation of the size of our project considering the functions that are meaningful for our software. In particular we will consider the following program characteristics called function types:

* Internal Logic Files
* External Interface Files
* External Input
* External Output
* External Inquiry

In order to have a good estimation we will use this table obtained with statistical procedures from real projects.

|  |  |  |  |
| --- | --- | --- | --- |
| FUNCTION TYPES | WEIGHT | | |
| **Simple** | **Medium** | **Complex** |
| Internal Logic Files | 7 | 10 | 15 |
| External Interface Files | 5 | 7 | 10 |
| External Inputs | 3 | 4 | 6 |
| External Outputs | 4 | 5 | 7 |
| External Inquiries | 3 | 4 | 6 |

1. INTERNAL LOGIC FILES (ILFs)

In this section I am going to list and analyse the most important Internal Logical File (ILF) of our application and so the homogeneous sets of data used and managed by the application.

First of all our system has to store all the information about clients and so the following fields: name, surname, mobile phone number, email address, password, code, driving licence, credit card and his state (Dismounted, Reserving, Driving, OnBreak). All this attributes are strings (except the state) and are stored in a single table. In a similar way also the assistants are structured even if they don’t have associated the driving licence, the credit card and the state.

Then another data managed by PowerEnjoy is the car which has several fields: code, model, battery, capacity which are stored as strings or integers. Then we have also the state of the car (Available, Reserved, OnCourse, OnBreak and NonAvailable) and two Boolean variable pointing out that the car is locked and in recharging state.

Then we store also the information about the town of Milan that are necessary for our application. So we consider the set of safeAreas which are identified by a code and by their coordinates. In a similar way also the parking stations are stored. However in this situation we must also save the information about rechargers which have a code and a Boolean that indicates if they are free or occupied.

Then there are the courses which are more complex than the previous one and so contains more attributes: the time of start and end, the data of start and end, the cash amount, the position of start and end, the discounts used, the client and the car involved and the correspondent reservation. All this field are stored in a single table.

Finally there are the reservations that must contains the client, the car and the course associated, and the information about the countdown.

All these sets of data have an extremely simple structure being composed of a small number of fields. The only exceptions are courses which are more structured and therefore can be considered of medium complexity.

The following table summarizes what has been said in this paragraph.

|  |  |  |
| --- | --- | --- |
| ILF | Complexity | FPs |
| Clients  Assistants  Cars  SafeAreas  ParkingStation  Courses  Reservations | Simple  Simple  Simple  Simple  Simple  Medium  Simple | 7  7  7  7  7  10  7 |
| Total | | 52 |

1. EXTERNAL INTERFACE FILES (ELFs)

With External Interface Files (EIF) we mean a user identifiable group of logically related data that is used for reference purposes only. This kind of data resides entirely outside the application boundary and is maintained by another application external inputs. The external interface file is an internal logical file for another application.

In our project the only external data sources are the following:

* The mapping service which manages all the information about the positions in term of latitude and longitude;
* The payment system which handles the payments that occur in the services offered by the application.

In both the situation the amount of data that is used is extremely small and so their complexity is considered simple.

|  |  |  |
| --- | --- | --- |
| EIF | Complexity | FPs |
| Map position  Paiments | Simple  Simple | 5  5 |
| Total | | 10 |

1. EXTERNAL INPUTS (EIs)

In this section we are going to consider all the typologies of interactions with external users of our application. In fact with External Input we mean data that are coming external to the application and may be used to maintain one or more internal logical files. In order to be clear the inputs will be divided according to the types of user.

Clients:

* Login/Logout: the client must insert his user name and the password in the proper form.
* Registration: the client must insert all required data which are immediately verified.
* Modify profile: the client after the login must insert the new credentials.
* Reserve a car: the client must insert his current position or a specific one and then has to select the car that he wants to pick up. He can also enable the saving option in order to find safe areas near the destination inserted. This is a slightly more structured operation and so can be considered of medium complexity.
* End course/Leave car in break: the client must answer if he wants to end is ride or leave the car in break.

Assistant:

* Login/Logout: it is the same situation as in the client
* Change car state: the assistant must insert the new state of the car. If the modified state in NonAvailable he must add the type of damage.

All these external inputs are extremely easy because they involve few inputs.

The first exceptions is client registration in which the user has to insert a lot of data that has to be verified and so it can be considered of medium complexity.

Then the reservation of a car is complex because there are interactions with several other entities such as car, course, client and safe area.

In the following table we show the function points associated with the External Inputs.

|  |  |  |  |
| --- | --- | --- | --- |
| EI | Complexity | | FPs |
| Client login/logout  Client registration  Modify profile  Reserve a car  End course / leave the car  Assistant login/logout  Change car state | | Simple  Average  Simple  Complex  Simple  Simple  Simple | 2x3  4  3  6  3  2x3  3 |
| Total | | | 31 |

1. EXTERNAL INQUIRIES (EQs)

In this section we are going to analyse the External Inquiries that occur in our application. They are elementary processes that send data or control information outside the application boundary in order to present data to a user through the retrieval of data or control information from an ILF or EIF.

In PowerEnjoy there are the following External Inquiries:

* A client can retrieve the actual reservation and the historical one.
* A client can retrieve the actual course and the historical one
* A client can retrieve all available cars near a position inserted or calculated by the GPS.
* A client can retrieve the sets of safe areas near a position inserted (this process occurs when he activates the saving option).
* An assistant can retrieve the list of cars and the information about their state.

All these operations can be carried out easily through query that extract data from a singular table and so can be considered of simple complexity.

|  |  |  |  |
| --- | --- | --- | --- |
| EQ | Complexity | | FPs |
| Retrieve reservation  Retrieve course  Retrieve car available near position  Retrieve safe areas near position  Retrieves cars and their state (by assistants) | | Simple  Simple  Simple  Simple  Simple | 3  3  3  3  3 |
| Total | | | 15 |

1. EXTERNAL OUTPUTS (EOs)

In this section we are going to examine External Output and so the elementary processes that send data or control information outside the application boundary. The primary purpose of external outputs is to present information to a user through processing logic other than, or in addition to the retrieval of data or control information.

The situation in which our system has to communicate to the user outside the context of an inquiry are the following:

* Notify the client that the registration has been completed successfully.
* Notify the client that the reservation has been accepted.
* Notify the client that the payment has been completed successfully.
* Notify the client that the car has been locked properly.
* Notify the client that the ReservationCountdown or the CourtesyCountdown expired
* Notify the assistant that the car state has been changed correctly.

All these operations are extremely easy because they involve only few data in the interactions with the external users and for this reason they can be considered of simple complexity.

In the following table the External Output are listed with the Function Points associated.

|  |  |  |  |
| --- | --- | --- | --- |
| EO | Complexity | | FPs |
| Registration completed notification  Reservation accepted notification  Payment completed notification  Car locked notification  Countdown expired notification  Car state changed notification | | Simple  Simple  Simple  Simple  Simple  Simple | 4  4  4  4  4  4 |
| Total | | | 24 |

1. OVERALL ESTIMATION

Finally in this last section we summarize the results obtained in the previous paragraphs. In particular the following table shows the function points associated with all function types

|  |  |
| --- | --- |
| Function type | FPs |
| Internal Logic Files  External Logic Files  External Inputs  External Inquiries  External Outputs | 52  10  31  15  24 |
| Total | 132 |

So in conclusion we have 132 Function Points that according to the following table can be converted in lines of code.

* High: SLOC = 132 \* 67 = 8844
* Median: SLOC = 132 \* 49 = 6468
* Average: SLOC = 132 \* 46 = 6072
* Low: SLOC = 132 \* 15 = 1980

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| LANGUAGE | SLOC/FP | | |  |
| **Average** | **Median** | **Low** | **High** |
| JEE2 | 46 | 49 | 15 | 67 |

* 1. COST AND EFFORT ESTIMATION: COCOMO II

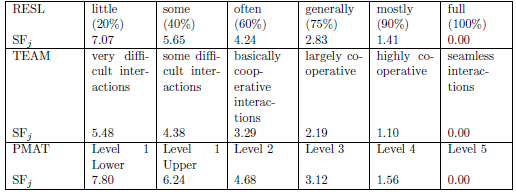
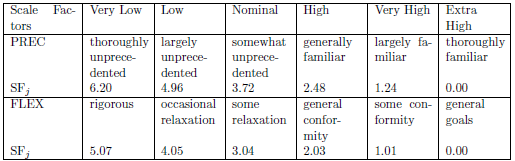
In this section we're going to estimate cost and effort needed to develop our project using COCOMO II

* + 1. SCALE DRIVERS

This is an analysis of our scale drivers:

* Precedentedness: this value reflects the experience of our developers in developing project similar to this product. The selected value is LOW, form the moment that we develop a large scale project for the first time.
* Development Flexibility: this value reflects how much our project in flexible in term of obligation to respect requirement and external constraint. The selected value is LOW, from the moment that we are forced to respect closely the requirement, even if we are quite free in the choice of technologies.
* Architecture/Risk Resolution: this value reflects the level of consciousness about what are the risks, the budget and the schedule for our project. The selected value is HIGH, from the moment that we performed an extensive analysis of risk and a schedule estimation.
* Team Cohesion: this value reflects how much stakeholders know each other and are able to work in a team. The selected value for our team VERY HIGH.
* Process Maturity: this value reflects the maturity of our software organization. The selected value is LEVEL 4, from the moment that we didn’t have particular problems in developing our project, although this is our first work of this type.

Below is reported the official COCOMO II table that assign a value for each scale factor



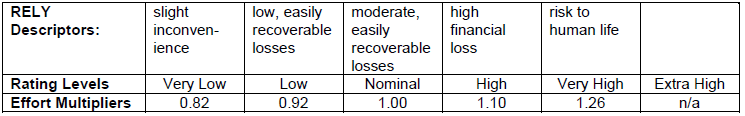
This is the result of our evaluation

|  |  |  |  |
| --- | --- | --- | --- |
| Scale Driver | Factor | | Value |
| Precedentedness (PREC)  Development flexibility (FLEX)  Risk resolution (RESL)  Team cohesion (TEAM)  Process maturity (PMAT) | | LOW  LOW  HIGH  VERY HIGH  LEVEL 4 | 4.96  4.05  2.83  1.10  1.56 |
| Total | | | 14.5 |

* + 1. COST DRIVERS

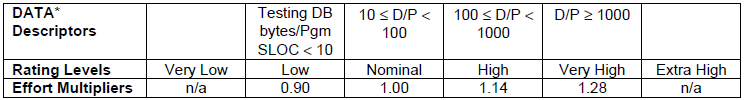
Required Software Reliability (RELY)

The system is a novelty and a great innovation for the city. This type of service aims to expand and reach most of the population so, in order to spread smoothly and avoid important financial losses, it must be enough reliable. The selected value is HIGH.



Data Base Size (DATA)

This measure tries to capture the effective size of our database. An estimation is to reach about 1 GB of database, distributed in an interval of SLOC which goes from about 2000 to about 9000. According to COCOMO II scale the selected value is NOMINAL.



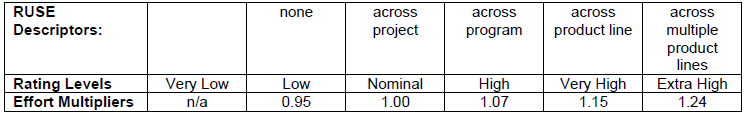
Product Complexity (CPLX)

According to the COCOMO II rating levels this value is set to HIGH.



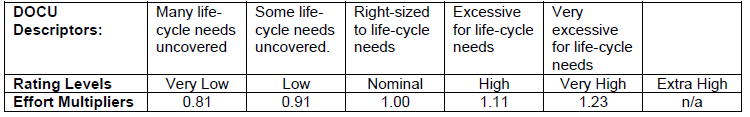
Developed for Reusability (RUSE)

Requirements do not impose external reusability for our software. Reusability is limited across project itself, so this value is set to NOMINAL.



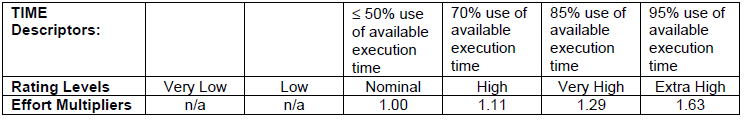
Documentation Match to Life-Cycle Needs (DOCU)

In our project documentation correctly reflects all needs in term of product life-cycle, so this value is set to NOMINAL.



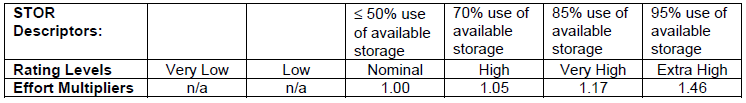
Execution Time Constraint (TIME)

The percentage of available execution time used by our application is estimated to be about 70%. So this value is set to HIGH.



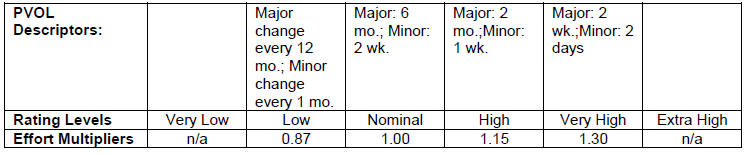
Main Storage Constraint (STOR)

The amount of storage usage with respect to the availability of the hardware is expected to be less than 50%. So this value is set to NOMINAL.



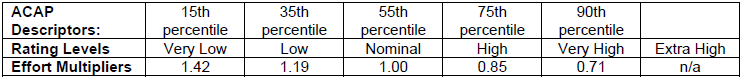
Platform Volatility (PVOL)

Our platform is fundamentally the mobile operating system. We estimate our client application to need a new release about once every 6 months. For this reason, this value is set to NOMINAL.



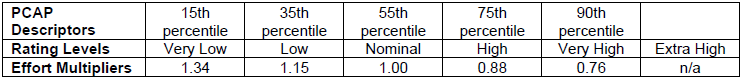
Analyst Capability (ACAP)

Our analyst team have a good skill in analysis of requirements and design of the software. They have faced the problem in a complete and accurate way, analysing all the problems of the real world. For this reason, this value is set to HIGH.



Programmer Capability (PCAP)

From the moment that the project has not yet been implemented this parameter is only an estimation. We think to have pretty good programming abilities, so this parameter is set to HIGH.



Personnel Continuity (PCON)

Our personnel are expected to be the same during the whole duration of the project. For this reason, this value is set to VERY HIGH.



Applications Experience (APEX)

Our project team don’t have a great experience in the development of application of this type, so this value is set to LOW.



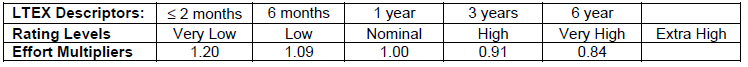
Platform Experience (PLEX)

This is our first project developed using Java EE platform. Our previous project didn’t include the usage of databases and networking. For this reason, this value is set to LOW.



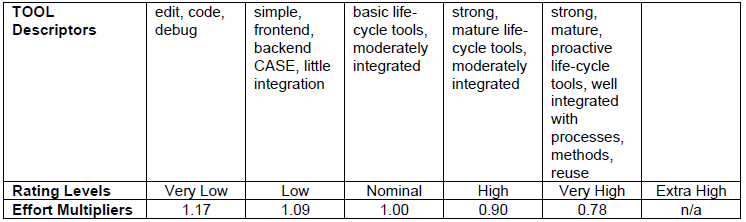
Language and Tool Experience (LTEX)

We have a good experience with Java programming language, but this is our first project developed using Java EE platform and tools of this kind. This value is set to NOMINAL.



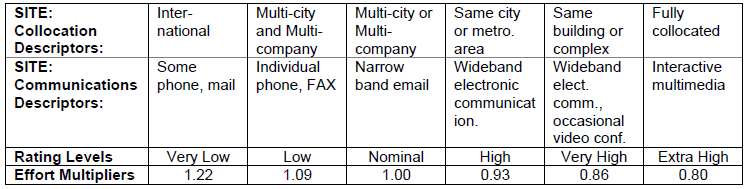
Use of Software Tools (TOOL)

We use for the development of our application moderately integrated life-cycle management tool, so the selected value is HIGH.



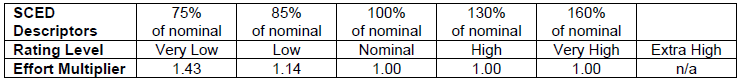
Multisite Development (SITE)

Even if members of our team live in the same city for the whole duration of the development of this project, we largely use internet services for communication and organization of the work (whatsapp, social network). For this reason, this value is set to HIGH.



Required Development Schedule (SCED)

In developing our project, we respect a nominal schedule for a project requiring an amount of effort of this kind, without particular stretch-out or acceleration. For this reason, the selected value is NOMINAL.



This is the result of our evaluation

|  |  |  |  |
| --- | --- | --- | --- |
| Cost Driver | Factor | | Value |
| Required Software Reliability (RELY)  Database size (DATA)  Product complexity (CPLX)  Required Reusability (RUSE)  Documentation match to life-cycle needs (DOCU)  Execution Time Constraint (TIME)  Main storage constraint (STOR)  Platform volatility (PVOL)  Analyst capability (ACAP)  Programmer capability (PCAP)  Personnel continuity (PCON)  Application Experience (APEX)  Platform Experience (PLEX)  Language and Tool Experience (LTEX)  Usage of Software Tools (TOOL)  Multisite development (SITE)  Required development schedule (SCED) | | HIGH  NOMINAL  HIGH  NOMINAL  NOMINAL  HIGH  NOMINAL  NOMINAL  HIGH  HIGH  LOW  LOW  LOW  NOMINAL  HIGH  HIGH  NOMINAL | 1.10  1.00  1.17  1.00  1.00  1.11  1.00  1.00  0.85  0.88  0.81  1.10  1.09  1.00  0.90  0.93  1.00 |
| Total | | | 0.8686 |

* + 1. EFFORT EQUATION

This is the final equation that estimate the effort in Person-Month (PM):

Effort = A \* \*

with:

A = 2.94 PM/KSLOC (Cocomo II constant)

= 0.72555 (product of cost drivers)

KSLOC = from 1980 to 8844 (derived from size estimation)

E = B + 0.01 \*

with:

B = 0.91 (Cocomo II constant)

*Lower bound (SLOC Low)*

Effort = 2.94 \* 0.8686 \* = 7676 ≈ 8PM

*Upper bound (SLOC High)*

Effort = 2.94 \* 0.8686 \* = 37229 ≈ 37PM

* + 1. SCHEDULE ESTIMATION

This is the equation that estimate the final schedule:

Duration = C \*

with:

C = 3.67

D = 0.28

E and B defined in the previous paragraph

*Lower bound (SLOC Low)*

Duration = 3.67 \* = 6.9 month

*Upper bound (SLOC High)*

Duration = 3.67 \* = 11.2 month

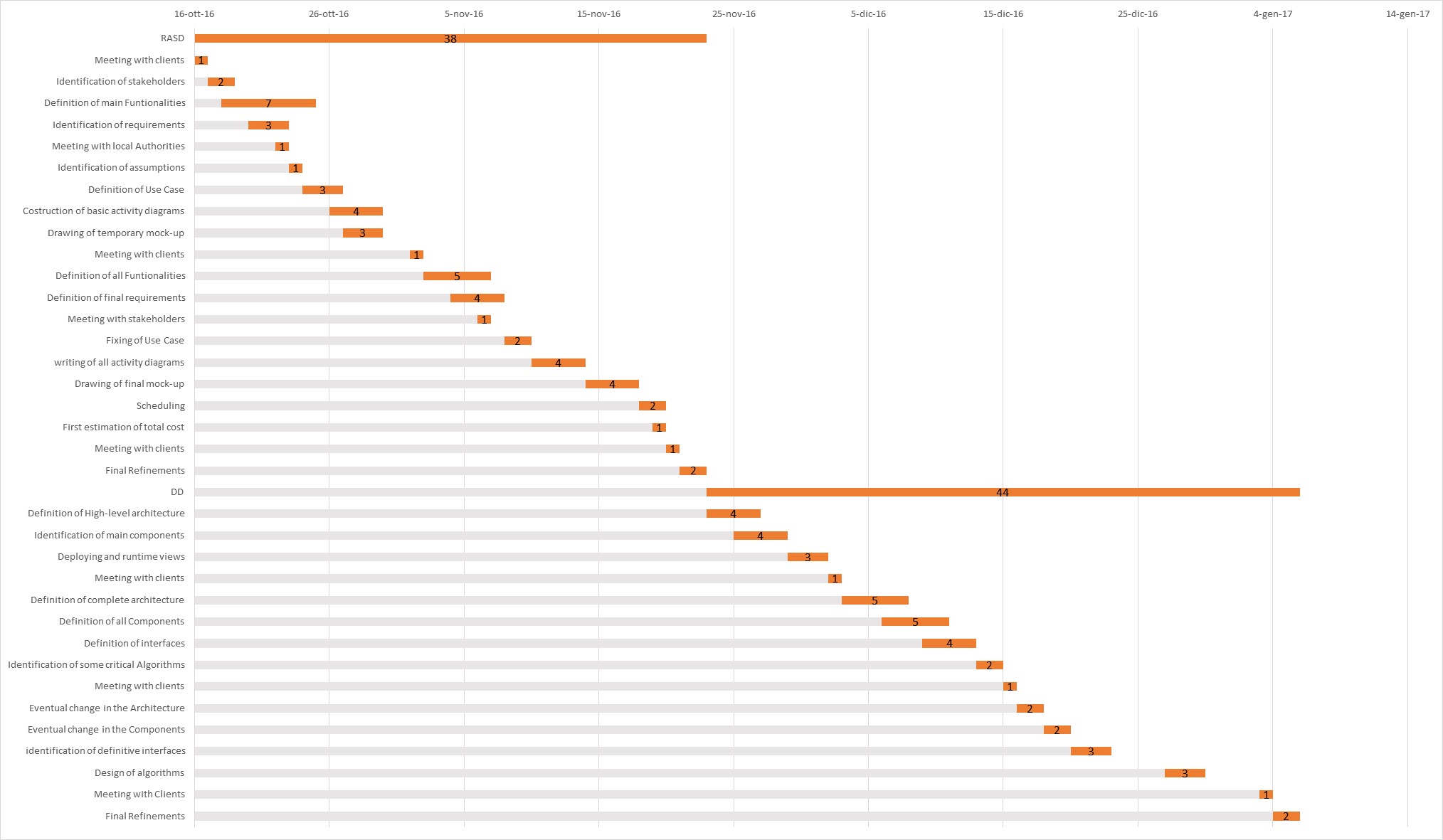
1. SCHEDULE

In this section we represent the schedule of our software in a time table. Every phase is represented with its amount of hours spent to be completed starting from the documentation of the software until its deployment.

The first part concerning the Requirements Analysis and Specifications Document, the Design Document and the Testing Document is treated in specific since they are documents already finished. However, the diagrams are treated as the project was developed in a real contest, so there are some tasks not affronted by us but that should be completed if the project was real (like the meetings).

The development part is a statistic prevision of the amount of hours that will be spent in each phase. During the development of the software the time table can change for some voice due for some unexpected aspects considered or events occurred in this phase.

Below there are two Gantt diagrams about the scheduling of our project. The first is about the documentation. The second is about the future development of our software. In the graph the sections are highlighted with all upper case label. The numbers written in the bars are the day spent for the activity.





1. RESOURCE ALLOCATION

In this section we are represented the different amounts of work for each team member. Many tasks were faced before all together and then finalized by one member. The amount of days for each team member are relative both for the team work and for individual work. Some more specific and realistic diagrams can be written during the progress in the development and deployment processes.

Even here the diagrams are split in two, to be more readable. The division of the first part reflect quite closely the true division of the work. For the second part the division are scheduled according to the different arguments treated by the different team members.

Diego Gaboardi

Giorgio Giardini

Riccardo Giol

1. RISK MANAGEMENT

In this section we evaluate and analyse some risks that can occur for our software. The risks can be technical, relative to the development of the software, relative to quality or timeliness ore relative to business aspects, like the market.

All the doubts relative to the administration of certain areas and management of some legal situations have to be discussed during the documentation with the authorities. The economic management, government grants and how these can change in the future have to be discussed. The economic warranty has to be discussed too. Some not calculated risk of this kind can brings important losses at the company.

Other legal risks can be linked with some changing in the legislation. The condition for a valid driver can change with another kind of driver licence or some other necessary data. This brings at an extension of the client data and so an upgrade of the system. The valid electric plug can change and so they have to be replaced as the power point of the cars. The cars can turn to be no more accordant to the law, and so all the cars have to be changed.

An economic risk can be the increasing of the cost of the electricity, and it can weight on the recharging of the cars. Other new taxes and increase of some goods can bring at a loss of util. so we always trying to keep a good margin in the income.

There are risks linked to some accidental events. All the accidents are managed by the warranty. If some client has a problem with a car, he can always call a green number and some assistants will help him. The assistants can manage many aspects linked to course, client, car and reservation and so solve any problems. If the problem is linked to some charging station the assistant can change the states of the station or temporary remove it, or the plug, from the system.

It Is even to evaluate the ambient in which the system will be inserted. In this ambient we have to monitor the concurrency systems, in order to provide a service that can be competitive in the market. We have to evaluate and exploit well the territory in which our system work, allocating in strategic and well diffused position the charging stations. This can be decisive for providing a well service to the client, and so to bring them to use it. The system has to be competitive especially in its debut. So for the first period some bonuses can be available for the new subscribers.

Some risks that can occur during the development and documentation of the software can be linked to the absence of an important figure in some specific sector. It is avoided by having always some personnel member that can hold the charge of the absent member. For how concern the publication times, all the deadlines have an error margin. The problem of the absence of some important figure can occur even during the utilization of the system, and it is avoided in the same way.

Problems can derivate from a changing in the interfaces with the external components. Any changing has to be presented at the company in advance, so the differences with the previous components can be managed. A no-managed changing can bring some dysfunctionalities in some services. Problems can occur even with the vendor of the tablet for the car and with the provider of its OS for the adjustment to do for well integrate it in the car system.

The loss of personal data and in particular of the source data can be disastrous. So all the data have to be replicate in different databases in different locations.