# Politecnico di Milano

Industrial and Information Engineering
Computer Science and Engineering

Software Engineering II Assignment
PowerEnJoy - car sharing

Academic Year: 2016/2017

Prof. Elisabetta Di Nitto

Alice Segato matr. 875045
Mark Edward Ferrer matr. 876650
Davide Bonacina matr. 876199

# PowerEnJoy

# Car Sharing App

# TABLE OF CONTENT

# LIST OF FIGURES

# 1 INTRODUCTION

## 1.1 Revision history

| Version | Date | Authors | Summary |
|---|---|---|---|
| 1.0 | 07/01/2017 | Mark Edward Ferrer, Alice Segato, Davide Bonacina | Initial release |

## 1.2 Purpose

The purpose of the Project Plan Document is to give guidelines for the definition of costs and effort the project requires. It emphasizes the main information for the budget estimation, the resource allocation, the scheduling of the activities and the risk management.

The document is divided in 4 sections:

- Project size, cost and effort estimation: detailed explanation of the cost and effort estimation, made using Function points and CO-COMO approaches, for the development phase in terms of lines of code.
- Schedule definition: propose of a possible schedule for all the phases of the project, which covers requirements analysis and implementation and testing.
- Resource allocation: assignment of the tasks between the different member of the development group
- Risk management: forecasting of the risks the development of the project could face

## 1.3 List of Definitions and Abbreviations

All the words defined in the RASD at section 1.5 are still valid and they will appear in this document too.

- DD: Design Document.
- RASD: Requirements analysis and Specification Document.
- DB: the database layer, handled by a DBMS.
- UI: User Interface.
- GUI: graphical user interface is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators;
- COCOMO: Constructive Cost Model is a procedural software cost estimation model.
- Function Point: unit of measurement to express the amount of business functionality an information system provides to a user. Function points are used to compute a functional size measurement of software. The cost of a single unit is calculated from past projects.

## 1.4 Reference Documents

- Assignment AA 2016-2017.pdf;
- Project planning example document.pdf.

# 2 Project size, cost and effort estimation

This section of the document provides an estimation about the size, the cost and the effort necessary for this project. On order to make a better estimation, we consider only the business logic code excluding the user interface code.

For the estimation of the project size, we use the Function Points approach that take in consideration all the features that the system provides to the user. This estimation is also related to the number of lines of code to be written in Java, useful also for effort estimation with COCOMO approach.

COCOMO approach relies on an equation based on different variables that has as basis the number of lines of code estimated in the Function Points section.

## 2.1 Size estimation: Function Points

Function points gives an estimation on the project size taking as input some symbols that represent the complexity of the functionality to rate. These symbols are Record Element Type (RET), Data Element Type (DET) and File Type Referenced (FTR) that contribute to the calculus of the complexity of each functionality of the system. The attribution of these symbols is made through the count of the types of records/data visible to the user, managed by each function. The complexity is assigned according to the tables below that take as entry these three variables.

For Internal Logic Files (ILF) and External Interface Files (EIF)

| | **Data Element Type (DET)** | | |
|---|---|---|---|
| **Record Element Type (RET)** | **1-19** | **20-50** | **51+** |
| **1** | Low | Low | Avg |
| **2-5** | Low | Avg | High |
| **6+** | Avg | High | High |

For External Output (EO) and External Inquiry (EQ)

| | **Data Element Type (DET)** | | |
|---|---|---|---|
| **File Type Referenced (FTR)** | **1-5** | **6-19** | **20+** |
| **1** | Low | Low | Avg |
| **2-3** | Low | Avg | High |
| **4+** | Avg | High | High |

For External Input (EI)

| | Data Element Type (DET) | | |
|---|---|---|---|
| **File Type Referenced (FTR)** | **1-4** | **5-15** | **16+** |
| **1** | Low | Low | Avg |
| **2-3** | Low | Avg | High |
| **4+** | Avg | High | High |

Unadjusted Function Points (UFP) Complexity Weights

| | Complexity Weight | | |
|---|---|---|---|
| **File Type** | **Low** | **Average** | **High** |
| **ILF** | 7 | 10 | 15 |
| **ELF** | 5 | 7 | 10 |
| **EI** | 3 | 4 | 6 |
| **EO** | 4 | 5 | 7 |
| **EQ** | 3 | 4 | 6 |

### *2.1.1 Internal Logic Files (ILF)*

According to the IFPUG, “An ILF is a user-identifiable group of logically related data or control information maintained within the boundary of the application. The primary intent of an ILF is to hold data maintained through one or more elementary processes of the application being counted”. Internal logic files are identifiable groups of logically related data that resides entirely within the application boundary and is maintained through External Inputs. They store any kind of data that will be used in a second moment for any purpose. All the data stored in the database are actually ILFs.

Users’ data are stored as Internal Logic Files in the main database including username, password, email, birth date, driving license number and payment method (which are the most relevant data needed for suitability about driving and consequently for reserving a car). In addition, name and surname are included in this table for communications in case of necessity. This logic file has 1 RET and 8 DET.

Administrators and employees’ table that is composed by username, password and position. This logical file has 1 RET and 3 DETs.

Cars’ data are stored in a separate table on the database and they include plate number as unique identifier, position (stored as a couple of two floating point numbers), car state (already defined in the RASD) and all the sensors’ information (number of passengers, battery charge, plug status and doors status, necessary for the car lock/unlock function). These data are useful for the business logic to track the cars, to maintain them and to compute a suitable parking destination (based on the destination inserted by the user) in case the user sets the money saving option. This logic file has 1 RET and 7 DET.

Also parking information are stored in the main database in a separate table: it contains Parking ID that is the unique identifier, parking type (it is a flag that distinguishes if a parking is a normal safe area or a special parking area), position, number of free slots, number of free plugs and number of available cars. These data are useful for better distribution of cars for the money saving option and for the reservation system that picks the best parking where users can take the car. This logic file has 1 RET and 6 DET.

Reservations are stored in a dedicated table on the main database. This table links the cars with the users and the attributes that compose it are reservation date, return date (this field is updated at the moment of the lock of the car or at the end of the pit stop timer, inserting the current timestamp), user, payment flag (that indicates if the reservation has been payed already) and money saving option flag. These are the main attributes used for the calculation of the drive price and the billing but there are also, start position, end position and duration of the drive for statistical purpose (used mainly to map the mostly used parking). This logic file has 3 RET and 21 DET.

Finally, the system keeps track of the user apps currently connected with a list of objects representing the clients on which it is working. This list is stored in the main memory to allow fast read and update of the list by the main system and it is useful for the observer design pattern to inform users about car states. This logic file has 1 RET and 1 DET.

According to the tables listed above the complexity of these ILFs is listed here:

| ILF | Complexity | FPs |
|---|---|---|
| User data | Low | 7 |
| Admin data | Low | 7 |
| Car data | Low | 7 |
| Parking data | Low | 7 |
| Reservation | Avg | 10 |
| Active clients | Low | 7 |
| **Total** | | 45 |

### *2.1.2 External Interface Files (EIF)*

According to the IFPUG, "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 simpler processes within the boundary of the application counted. This means an EIF counted for an application must be in an ILF in another application".

External Interface Files represent the data that the system will use/reference, but data that are not maintained by the system. This means that an EIF counted for an application must be in an ILF in another application.

Since they represent data handled by the main system but produced by another component (that are the user app and the car computer), all the information about GPS mapping on both user and car mobile devices and sensor detection on the car are the only EIFs of the main system.

The interactions that involve the EIFs are:

- Visualization of nearest parking (2 RETs that are user table and parking table, 2 DETs that are position attributes on both tables);
- Generation of shortest path to the selected parking for pickup (same as the previous interaction);
- Computation of the destination parking for money saving option (4 RETs that are all 4 the main tables in the database, 6 DETs that are user and money saving option in reservation, user position, car position and parking position);
- Retrieval of the car in case of parking outside of a safe area (caused by end of pit stop timer or car failure) (1 RET that is the car table and 1 DET that is the car position);
- Computation of extra fees or discounts based on the sensors' data (4 RETs that are all 4 the main tables in the database and 6 DETs that are, car position and sensors' data, parking position, parking plug slots and reservation money saving option flag);
- Calculation of the final amount to pay intended as the basic tariff (2 RET that are user and reservation tables, and 6 DETs that are user's payment method, reservation user, reservation date, return date and the two flags).

The table below represents the FPs assigned for these EIFs:

| EIF | Complexity | FPs |
|---|---|---|
| Nearest parking | Low | 5 |
| Navigator to the selected parking | Low | 5 |
| Money saving option | High | 10 |
| Car retrieval | Low | 5 |
| Extra fees and discounts | High | 10 |
| Price calculation | Low | 5 |
| **Total** | | 40 |

## *2.1.3 External Inputs (EI)*

According to the IFPUG, External Inputs "are elementary processes that process 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 behavior of the system". All user inputs and all file feeds are external input for the main system.

Since there are mainly 3 types of user and one of them can send two different types of input, we clamp all the input types in 4 categories:

1. User through user app:
    a. Login/registration: they involve 1 FTR each and respectively 8 and 2 DETs (in this final case, the DETs are username and password);
    b. User info visualization: it involves the complete user table (1 FTR, 8 DETs);
    c. Car finding: it involves user, car and parking tables modifying or maintaining user position, car position, ID and state, parking position, number of available cars (3 FTRs, 6 DETs);
    d. Car reservation: it involves all the tables and especially the attributes username and position of the user table, car position and car state, parking position and number of available cars and all the attributes of the reservation table, since it is a new tuple insertion (4 FTRs, 11 DETs);
    e. Money saving option activation (MSO activation): it involves user, parking and reservation tables, on user position, parking position and reservation money saving option flag attributes (3 FTRs, 3 DETs);
    f. Car lock/unlock: they involve only the car table and modifies the car state attribute (1 FTR, 1 DET each);
2. User through car computer:
    a. Car pit stop/full stop: they involve respectively the car table and both car and reservation tables. It handles the attributes car state and reservation return date (1 FTR and 1 DET, 2 FTRs and 2 DETs);
3. Administrator through admin app:
    a. Removal of a user: it involves the user and the reservation tables and all their attributes but also the user app list (3 FTRs and 14 DETs);
    b. Add/remove a parking: it involves the parking table with all his attributes (1 FTR, 6 DETs each);
    c. Add/remove car: it involves the car and parking tables with all car attributes and parking's number of available cars and free slots (2 FTR, 8 DETs each);
    d. Car maintenance: it involves the car and the parking tables modifying the car state, the parking number of free slots and number of available cars (2 FTRs, 3 DETs);
4. Employee through employee app:
    a. Car retrieval: it involves the admin and car table, using the admin position and the car position (2 FTRs, 2 DETs).

The final calculation is listed below:

| EI | Complexity | FPs |
|---|---|---|
| Login/Registration | Low | 3x2 |
| User info | Low | 3 |
| Car finding | Avg | 4 |
| Car reservation | High | 6 |
| MSO activation | Low | 3 |
| Lock/unlock | Low | 3x2 |
| Pit stop/full stop | Low | 3x2 |
| Remove user | Avg | 4 |
| Add/remove parking | Low | 3 |
| Add/remove car | Avg | 4 |
| Car maintenance | Low | 3 |
| Car retrieval | Low | 3 |
| **Total** | | 51 |

### *2.1.4 External Inquiries (EQ)*

According to the IFPUG, "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". They represent all the data queries made by a user, that can be a common user or a privileged user like admins or employees.

There are 3 interactions that can be seen as External Inquiries:

- User information visualization explained before;
- Car info visualization by admin for maintenance;
- Car position visualization by employee for retrieval.

They are very simple so they have low complexity and they reference 1 RTF (respectively user table and car table for the last two) and all the needed attributes of each table.

| EQ | Complexity | FPs |
|---|---|---|
| User information | Low | 3 |
| Car information | Low | 3 |
| Car position | Low | 3 |
| **Total** | | 9 |

### *2.1.5 External Outputs (EO)*

According to the IFPUG, "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".

These are all very simple actions because they need to notify the user that something has happened so in the final table they will have all low complexity:

- Send the verification email;
- Notify the user that the registration was successful or not;
- Notify the user that the reservation was successful or not;
- Notify the user that the payment was successful or not;
- Notify the admin that the car add was successful or not;
- Notify the admin that the parking add was successful or not;
- Notify the employee that there's a car to retrieve;

| EO | Complexity | FPs |
|---|---|---|
| Verification email | Low | 4 |
| Registration notification | Low | 4 |
| Reservation notification | Low | 4 |
| Payment notification | Low | 4 |
| Car add notification | Low | 4 |
| Payment add notification | Low | 4 |
| Retrieval notification | Low | 4 |
| **Total** | | 28 |

### 2.1.6 Overall Estimation

This is the resulting table that summarize the computation of the FP:

| Function Type | Value |
|---|---|
| Internal Logic Files | 45 |
| External Interface Files | 40 |
| External Inputs | 51 |
| External Inquiries | 9 |
| External Outputs | 28 |
| **Total** | 173 |

Considering the fact that the system will be implemented in Java Enterprise Edition (J2E or JEE) without considering the implementation of the different applications for both common users and privileged users (that can be implemented only as presentation layer with no business logic) the lines of code will be between $B_L$ and $B_U$ where:

$$B_L = FP * 46 = 7958$$

$$B_U = FP * 67 = 11591$$

## 2.2 Cost and effort estimation: COCOMO II

In this section we're going to use the COCOMO II approach to estimate the cost and effort needed to develop PowerEnjoy.

### 2.2.1 Scale Drivers

Exponent E is the aggregation of five scale factors that account for some possible overheads encountered for software projects. Each factor has a range of rating levels, from Very low to Extra high, and each rating level has a weight. The specific value of the weight is called a scale factor (SF).

In order to evaluate the values of the scale drivers, we refer to the following official COCOMO II table:

| Scale Factor | Very Low | Low | Nominal | High | Very High | Extra |
|---|---|---|---|---|---|---|
| **PREC** | Thoroughly unprece-dented | largely unprece-dented | somewhat unprece-dented | generally familiar | largely familiar | thoroughly familiar |
| SFj | 6.20 | 4.96 | 3.72 | 2.48 | 1.24 | 0.00 |
| **FLEX** | rigorous | occasional relaxation | some relaxation | general conformity | some conformity | general goals |
| SFj | 5.07 | 4.05 | 3.04 | 2.03 | 1.01 | 0.00 |
| **RESL** | little (20%) | some (40%) | often (60%) | generally (75%) | mostly (90%) | full (100%) |
| SFj | 7.07 | 5.65 | 4.24 | 2.83 | 1.41 | 0.00 |
| **TEAM** | very difficult interactions | some difficult interactions | basically cooperative interactions | largely cooperative | highly cooperative | seamless interactions |
| SFj | 5.48 | 4.38 | 3.29 | 2.19 | 1.10 | 0.00 |
| **PMAT** | Level 1 Lower | Level 1 Upper | Level 2 | Level 3 | Level 4 | Level 5 |
| SFj | 7.80 | 6.24 | 4.68 | 3.12 | 1.56 | 0.00 |

### Precedentedness:

Reflects the previous experience of the organization with this type of project. Very low means no previous experience, Extra high means that the organization is completely familiar with this application domain. The precedentedness is LOW because we have some experience of software design but most of the notions used in this project are new to us.

### Development flexibility:

Reflects the degree of flexibility in the development process. Very low means a prescribed process is used; Extra high means that the client only sets general goals. We set it to NOMINAL because we have to follow a prescribed process, but we had a certain degree of flexibility in the definition of the requirements and in the design process.

### Risk resolution:

Reflects the extent of risk analysis carried out. Very low means little analysis, Extra high means a complete a thorough risk analysis. We set it to HIGH, because many reason (a rather detailed risk analysis is carried out in chapter 5).

### Team cohesion:

Reflects how well the development team know each other and work together. Very low means very difficult interactions, very high means an integrated and effective team with no communication problems. We set it to VERY HIGH, since the cohesion among the three of us is optimal.

### Process maturity:

Although we had some problems during the development of the project, the goals have been successfully achieved. Since this is our first project of this kind, this value is set to HIGH (Level 3).

The results of our evaluation is the following:

| Scale Driver | Factor | Value |
|---|---|---|
| Precedentedness (PREC) | LOW | 4.96 |
| Development flexibility (FLEX) | NOMINAL | 3.04 |
| Risk resolution (RESL) | HIGH | 2.83 |
| Team cohesion (TEAM) | VERY HIGH | 1.10 |
| Process maturity (PMAT) | HIGH | 3.12 |
| Total | | 1.0605 |

### *2.2.2 Cost Drivers*

The parameter EAF is derived from the effort multipliers (EM) of the Cost drivers. Cost drivers are used to capture characteristics of the product under development, of the personnel working on it, and of general practices that affect the effort to complete the project.

#### Required Software Reliability:

This is the measure of the extent to which the software must perform its intended function over a period of time. If the effect of a software failure is only slight inconvenience, then reliability is low. If a failure would risk human life, then reliability is very high.
Our value will be LOW because a failure doesn't have critical consequences.

| RELY Cost Drivers | | | | | | |
|---|---|---|---|---|---|---|
| **RELY Descriptors** | slightly inconvenience | easily recoverable losses | moderate recoverable losses | high financial loss | risk to human life | |
| **Rating level** | Very low | Low | Nominal | High | Very High | Extra High |
| **Effort multipliers** | 0.82 | 0.92 | 1.00 | 1.10 | 1.26 | n/a |

#### Database size:

When you design a database, you may have to estimate how large the database will be when filled with data. Estimating the size of the database can help you determine the hardware configuration you will require to do the following: Achieve the performance required by your applications. Guarantee the appropriate physical amount of disk space required to store the data and indexes. Estimating the size of a database can also help you determine whether the database design needs refining. For example, you may determine that the estimated size of the database is too large to implement in your organization and that more normalization is required. Conversely, the estimated size may be smaller than expected. This would allow you to denormalize the database to improve query performance. At this stage there is no testing database, but considering the size of the project the DATA cost driver being HIGH.

| DATA Cost Drivers | | | | | | |
|---|---|---|---|---|---|---|
| **DATA Descriptors** | | $(D/P) < 10$ | $10 \leq (D/P) \leq 100$ | $100 \leq D/P \leq 1000$ | $D/P > 1000$ | |
| **Rating level** | Very low | Low | Nominal | High | Very High | Extra High |
| **Effort multipliers** | n/a | 0.90 | 1.00 | 1.14 | 1.28 | n/a |

### Product complexity:

Complexity is divided into five areas: control operations, computational operations, device- dependent operations, data management operations, and user interface management operations. After considering the size and nature of PowerEnJoy system, we set this parameter to HIGH.

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

### Required reusability:

This cost driver accounts for the additional effort needed to construct components intended for reuse on the current or future projects. This effort is consumed with creating more generic design of software, more elaborate documentation, and more extensive testing to ensure components are ready for use in other applications. Since we require extensive documentation and thorough testing, it is reasonable to set this parameter to high but in our case, the reusability requirements are limited in scope to the project itself, so the RUSE cost driver is set to NOMINAL.

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

## Documentation match to life-cycle needs:

The rating scale for the DOCU cost driver is evaluated in terms of the suitability of the project's documentation to its lifecycle needs. We believe that documentation is crucial in any project, therefore we set this parameter to HIGH.

| DOCU Cost Driver | | | | | | |
|---|---|---|---|---|---|---|
| **DOCU Descriptors** | Many lifecycle needs uncovered | Some lifecycle needs uncovered | Rightsized to lifecycle needs | Rightsized to lifecycle needs | Very excessive for lifecycle needs | |
| **Rating level** | Very low | Low | Nominal | High | Very High | Extra High |
| **Effort multipliers** | 0.82 | 0.92 | 1.00 | 1.10 | 1.26 | n/a |

## Execution time constraint:

This is a measure of the execution time constraint imposed upon a software system. The expected amount of CPU used by PowerEnJoy system is estimated to be more than the 85% for this reason we decided to set this value HIGH.

| TIME Cost Driver | | | | | | |
|---|---|---|---|---|---|---|
| **TIME Descriptors** | | | ≤ 50% use of available execution time | 70% use of available execution time | 85% use of available execution time | 95% use of available execution time |
| **Rating level** | Very low | Low | Nominal | High | Very High | Extra High |
| **Effort multipliers** | n/a | n/a | 1.00 | 1.11 | 1.29 | 1.63 |

## Storage constraint:

This rating represents the degree of main storage constraint imposed on a software system or subsystem. In our project this parameter is not relevant because we will use less the 50% of use of available storage we decided to set it NOMINAL.

| STOR Cost Driver | | | | | | |
|---|---|---|---|---|---|---|
| **STOR Descriptors** | | | ≤ 50% use of available storage | 70% use of available storage | 85% use of available storage | 95% use of available storage |
| **Rating level** | Very low | Low | Nominal | High | Very High | Extra High |
| **Effort multipliers** | n/a | n/a | 1.00 | 1.05 | 1.17 | 1.46 |

## Platform Volatility:

The term platform is used here to mean the complex of hardware and software (OS, DBMS, etc.) the software product calls on to perform its tasks. Since the platform shouldn't change too often, this value is set to NOMINAL.

| PVOL Cost Driver | | | | | | |
|---|---|---|---|---|---|---|
| **PVOL Descriptors** | | Major change every 12 mo., minor change every 1 mo. | Major: 6mo; minor: 2wk. | Major: 2mo, minor: 1wk | Major: 2wk; minor: 2 days | |
| **Rating level** | Very low | Low | Nominal | High | Very High | Extra High |
| **Effort multipliers** | n/a | n/a | 1.00 | 1.05 | 1.17 | 1.46 |

## Analyst Capability:

Analysts are personnel who work on requirements and design. The major attributes that should be considered in this rating are analysis and design ability, efficiency and thoroughness, and the ability to communicate and cooperate. This parameter is set to HIGH, since we dedicated a great effort in analyzing the problem requirements and its potential integration in the real world, and also in designing the whole system.

| ACAP Cost Driver | | | | | | |
|---|---|---|---|---|---|---|
| **ACAP Descriptors** | 15th percentile | 35th percentile | 55th percentile | 75th percentile | 90th percentile | |
| **Rating level** | Very low | Low | Nominal | High | Very High | Extra High |
| **Effort multipliers** | 1.42 | 1.19 | 1.00 | 0.85 | 0.71 | n/a |

## Programmer Capability:

Evaluation should be based on the capability of the programmers as a team rather than as individuals. Major factors which should be considered in the rating are ability, efficiency and thoroughness, and the ability to communicate and cooperate. Should development be upon us, we set this parameter to HIGH.

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

## Application Experience:

The rating for this cost driver depends on the level of applications experience of the project team developing the software system. The ratings are defined in terms of the project team's equivalent level of experience with this type of application. Considering our education, we can set this parameter to <u>NOMINAL</u>.

| **APEX Cost Driver** | | | | | | |
|---|---|---|---|---|---|---|
| **APEX Descriptors** | ≤ 2 months | 6 months | 1 year | 3 years | 6 years | |
| **Rating level** | Very low | Low | Nominal | High | Very High | Extra High |
| **Effort multipliers** | 1.22 | 1.10 | 1.00 | 0.88 | 0.81 | n/a |

## Platform Experience:

The Post-Architecture model broadens the productivity influence of platform experiences, recognizing the importance of understanding the use of more powerful platforms, including more graphic user interface, database, networking, and distributed middleware capabilities. Our value is <u>VERY LOW</u> because of our few experiences

| **PLEX Cost Driver** | | | | | | |
|---|---|---|---|---|---|---|
| **PLEX Descriptors** | ≤ 2 months | 6 months | 1 year | 3 years | 6 years | |
| **Rating level** | Very low | Low | Nominal | High | Very High | Extra High |
| **Effort multipliers** | 1.19 | 1.09 | 1.00 | 0.91 | 0.85 | n/a |

## Language and Tool Experience:

This is a measure of the level of programming language and software tool experience of the project team developing the software system or subsystem. Software development includes the use of tools that perform requirements and design representation and analysis, configuration management, document extraction, library management, program style and formatting, consistency checking, planning and control, etc. In addition to experience in the project's programming language, experience on the project's supporting tool set also affects development effort. Considering our education, we can set this parameter to <u>NOMINAL</u>.

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

## Personnel continuity:

The rating scale for this factor is in terms of the project's annual personnel turnover. This parameter is set to VERY HIGH, since in our case the available time is less than 6% of the year.

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

## Usage of Software Tools:

This parameter evaluates the use of tools to support the development and testing. Since we set no mandatory prescription to the developers, we set this parameter to NOMINAL.

| TOOL Cost Driver | | | | | | |
|---|---|---|---|---|---|---|
| **TOOL Descriptors** | edit, code, debug | simple, frontend, backend CASE, little integration | basic life-cycle tools, moderately integrated | strong, mature lifecycle tools, moderately integrated | strong, mature, proactive lifecycle tools, well integrated with processes, methods, reuse | |
| **Rating level** | Very low | Low | Nominal | High | Very High | Extra High |
| **Effort multipliers** | 1.17 | 1.09 | 1.00 | 0.90 | 0.78 | n/a |

## Multisite development:

Given the increasing frequency of multisite developments, and indications that multisite development effects are significant, this parameter measures the impact on the development process of site collocation and communication support. Since we met daily and used phone calls and messaging applications to communicate, we set this parameter to EXTRA HIGH.

| SITE Cost Driver | | | | | | |
|---|---|---|---|---|---|---|
| **SITE Collocation Descriptors** | Interna-tional | Multicity and multi-company | Multicity or multi-company | Same city or metro area | Same building or complex | Fully collocated |
| **SITE Communications Descriptors** | Some phone, mail | Individual phone, fax | Narrow band email | Wideband electronic communication | Wideband elect. comm., occasional video conf. | Interactive multimedia |
| **Rating level** | Very low | Low | Nominal | High | Very High | Extra High |
| **Effort multipliers** | 1.22 | 1.09 | 1.00 | 0.93 | 0.86 | 0.80 |

## Required development schedule:

This rating measures the schedule constraint imposed on the project team developing the software. The ratings are defined in terms of the percentage of schedule stretch-out or acceleration with respect to a nominal schedule for a project requiring a given amount of effort. In spite of the well-defined deadlines, which facilitated the distribution of our efforts over the time, some phases required much work. For this reason, this parameter is set to HIGH.

| SCED Cost Driver | | | | | | |
|---|---|---|---|---|---|---|
| **SCED Descriptors** | 75% of nominal | 85% of nominal | 100% of nominal | 130% of nominal | 160% of nominal | |
| **Rating level** | Very low | Low | Nominal | High | Very High | Extra High |
| **Effort multipliers** | 1.43 | 1.14 | 1.00 | 1.00 | 1.00 | n/a |

Overall, our results are expressed by the following table:

| Cost Driver | Factor | Value |
|---|---|---|
| Required Software Reliability (RELY) | LOW | 0.92 |
| Database size (DATA) | HIGH | 1.14 |
| Product complexity (CPLX) | HIGH | 1.17 |
| Required Reusability (RUSE) | NOMINAL | 1.00 |
| Documentation match to life- cycle needs (DOCU) | HIGH | 1.10 |
| Execution Time Constraint (TIME) | HIGH | 1.29 |
| Main storage constraint (STOR) | NOMINAL | 1.00 |
| Platform volatility (PVOL) | NOMINAL | 1.00 |
| Analyst capability (ACAP) | HIGH | 0.85 |
| Programmer capability (PCAP) | HIGH | 0.88 |
| Application Experience (APEX) | NOMINAL | 1.00 |
| Platform Experience (PLEX) | VERY LOW | 1.19 |
| Language and Tool Experience (LTEX) | NOMINAL | 1.00 |
| Personnel continuity (PCON) | VERY LOW | 1.20 |
| Usage of Software Tools (TOOL) | NOMINAL | 1.00 |
| Multisite development (SITE) | EXTRAHIGH | 0.80 |
| Required development schedule (SCED) | HIGH | 1.00 |
| Total | | 1,48792 |

### *2.2.3 Effort equation*

#### Sizing the project

A reasonable size estimate of a project is very important for a good model estimation. However, at this stage this is a challenging operation, since it is nearly impossible to correctly guess how much of PoweEnJoY code will be newly written, and how much, instead, will be reused or readapted. That is why we are going to provide the (very unlikely) worst case estimation: we assume that the whole code will be written from scratch. The quantity s is the estimated number of source lines of code (SLOC):

$$s = p * FP$$

In our case the sizing of the project results:

$$\begin{cases} s = 46 * 173 = 7958 & (for\ lower\ bound\ the\ one\ we\ choose\ to\ evaluate\ our\ project\ in\ the\ worst\ case) \\ s = 67 * 173 = 11591 & (for\ the\ upper\ bound) \end{cases}$$

#### Effort estimation

Effort is expressed in terms of person-months (PM). A person-month is the amount of time one person spends working on the software development project for one month. COCOMO defines the following formula to estimate it:

$$e = 2.94 * (s/1000)^{E} * EAF$$

Exponent E and parameter EAF are defined as follows:

$$E = 0.91 + 0.01 * \sum_{1}^{5} Sfj$$

$$EAF = \prod_{i}^{n} EM_i$$

In the previous paragraphs we presented the factors from which exponent E and parameter EAF are derived.

Thanks to the results in equations E and EAF, we are now able to estimate the value of the effort, defined in equation e.

$$E = 1.0605$$

$$EAF = 1{,}48792$$

$$e = 2.94 * (7958/1000)^{1.06} * 1{,}48792 = 26{,}04$$

So, according to COCOMO II model and our estimation, almost exactly 27 person-months are needed to fully develop PowerEnJoy system.

### *2.2.4 Schedule estimation*

Now, thanks to the previous results, it is possible to evaluate the time to develop (t). Time to develop is the calendar time in months that goes from the determination of the product's requirements to the completion of an acceptance activity certifying that the product satisfies its requirements.

The time to develop is given by this formula:

$$t = 3.67 * (e)^{F}$$

Exponent F is defined as follows:

$$F = 0.28 + 0.2 * (E - 0.91)$$

By substituting the results from previous sections in equations, we obtain:

$$F = 0.28 + 0.2\ \ (1.0605 - 0.91) = 0.3101$$

$$t = 3.67 * (26{,}04)^{0.3101} = 10.08$$

The development of PowerEnJoy as a whole is expected to take more than 10.08 months.

By dividing the effort e by the time to develop t, we get the estimated number of people needed for the project:

$$n = e/t\ = 26{,}04/10.08 = 2.58$$

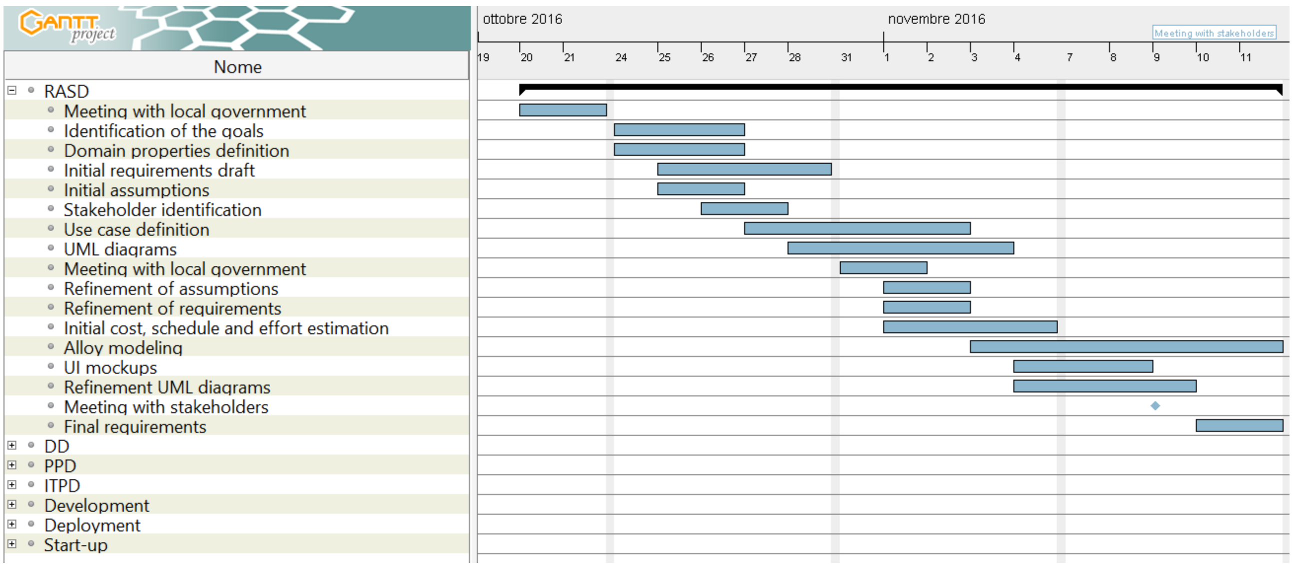
# 3 Schedule

The following detailed project schedule is the representation of the organization of every task of the project. It aims to give a general guidance for the project phases. Other schedules, that explains more in depth the phases, are expected to be juxtaposed to this one, during the project.

This schedule does not show how to the actual activities were organized by our team during the drawing up of the documents, but how they would have been in a real life project planning situation.

We decided to divide the schedule for each phase in order to guarantee a better readability.

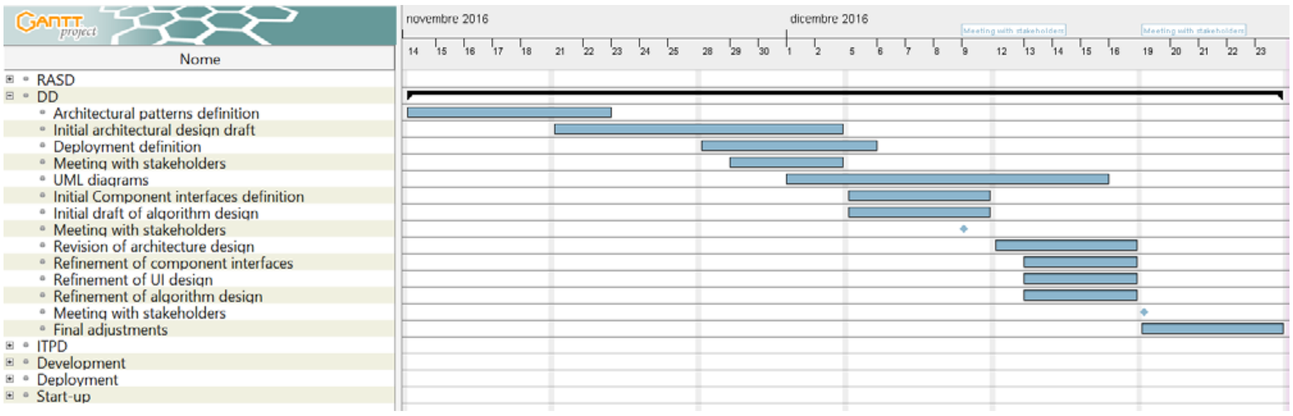
## 3.1 Requirements Analysis and Specification Document Schedule



Nome
- RASD
  - Meeting with local government
  - Identification of the goals
  - Domain properties definition
  - Initial requirements draft
  - Initial assumptions
  - Stakeholder identification
  - Use case definition
  - UML diagrams
  - Meeting with local government
  - Refinement of assumptions
  - Refinement of requirements
  - Initial cost, schedule and effort estimation
  - Alloy modeling
  - UI mockups
  - Refinement UML diagrams
  - Meeting with stakeholders
  - Final requirements
- DD
- PPD
- ITPD
- Development
- Deployment
- Start-up

*Figure 1*

## 3.2 Design Document Schedule



Nome
- RASD
- DD
  - Architectural patterns definition
  - Initial architectural design draft
  - Deployment definition
  - Meeting with stakeholders
  - UML diagrams
  - Initial Component interfaces definition
  - Initial draft of algorithm design
  - Meeting with stakeholders
  - Revision of architecture design
  - Refinement of component interfaces
  - Refinement of UI design
  - Refinement of algorithm design
  - Meeting with stakeholders
  - Final adjustments
- ITPD
- Development
- Deployment
- Start-up

*Figure 2*

## 3.3 Integration Testing Planning Schedule

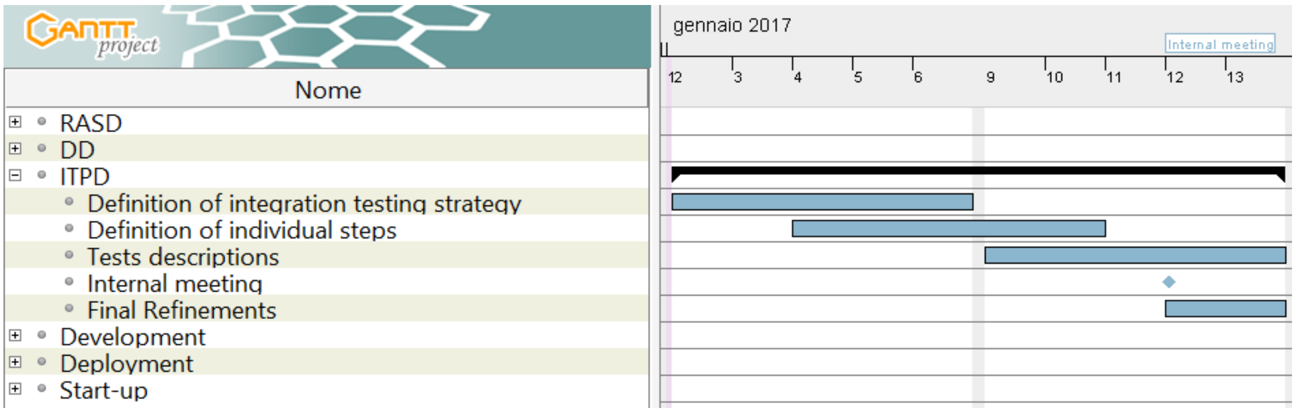


Figure 3

## 3.4 Development Schedule

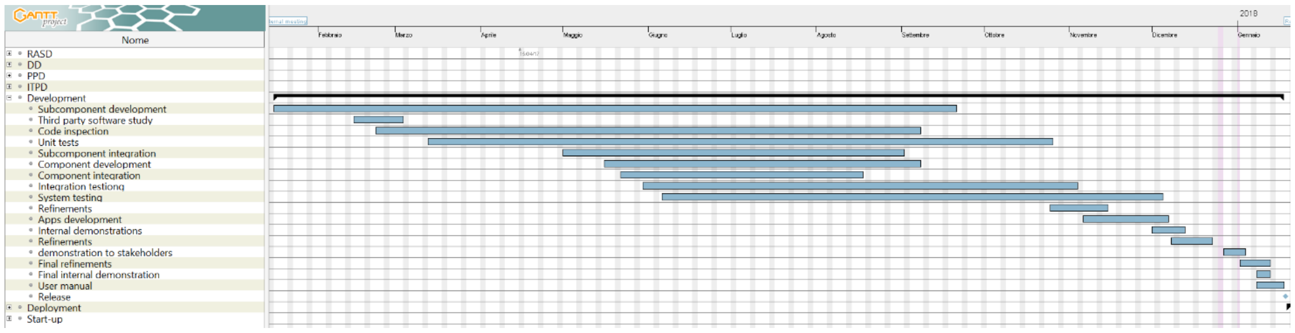


Figure 4

## 3.5 Deployment Schedule

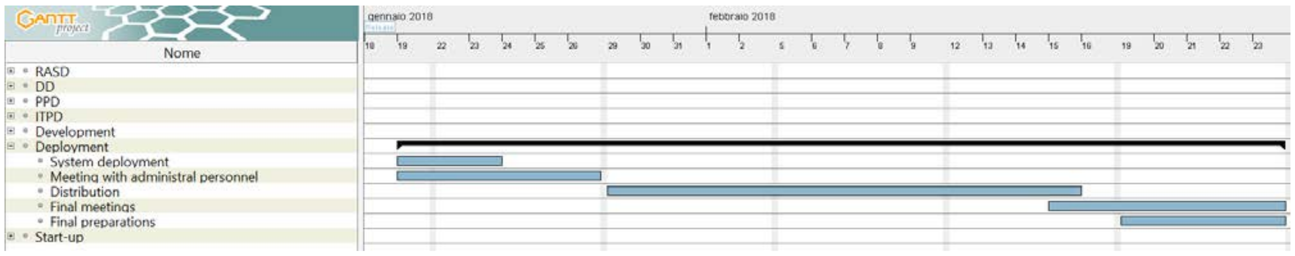


Figure 5

## 3.6 Start-up Schedule

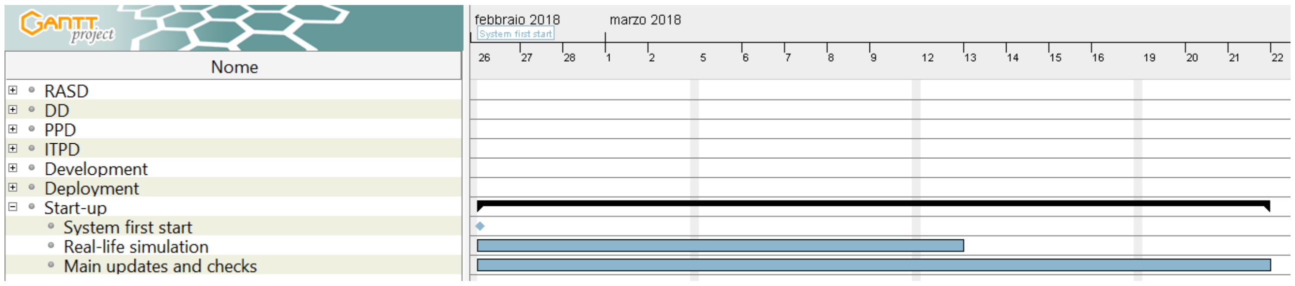


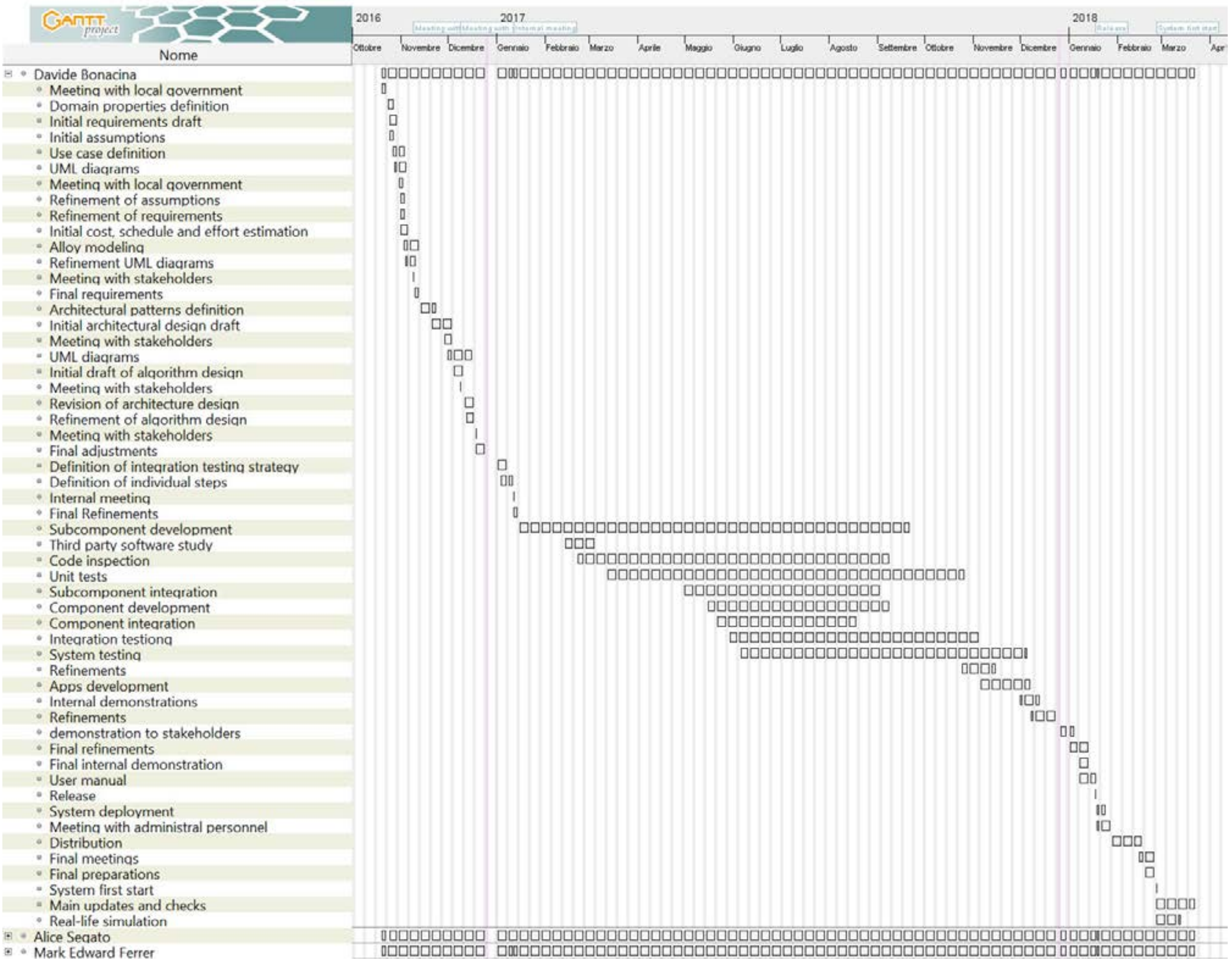
Figure 6

# 4 Resource Allocation

The following resource allocation schedule represent the division of the tasks presented in the previous section between the members of the team.

As the previous schedule, this does not reflect the exact division and effort spent on each task that members put during this project, but how they would have been divided in a real case situation.

## 4.1 Davide Bonacina

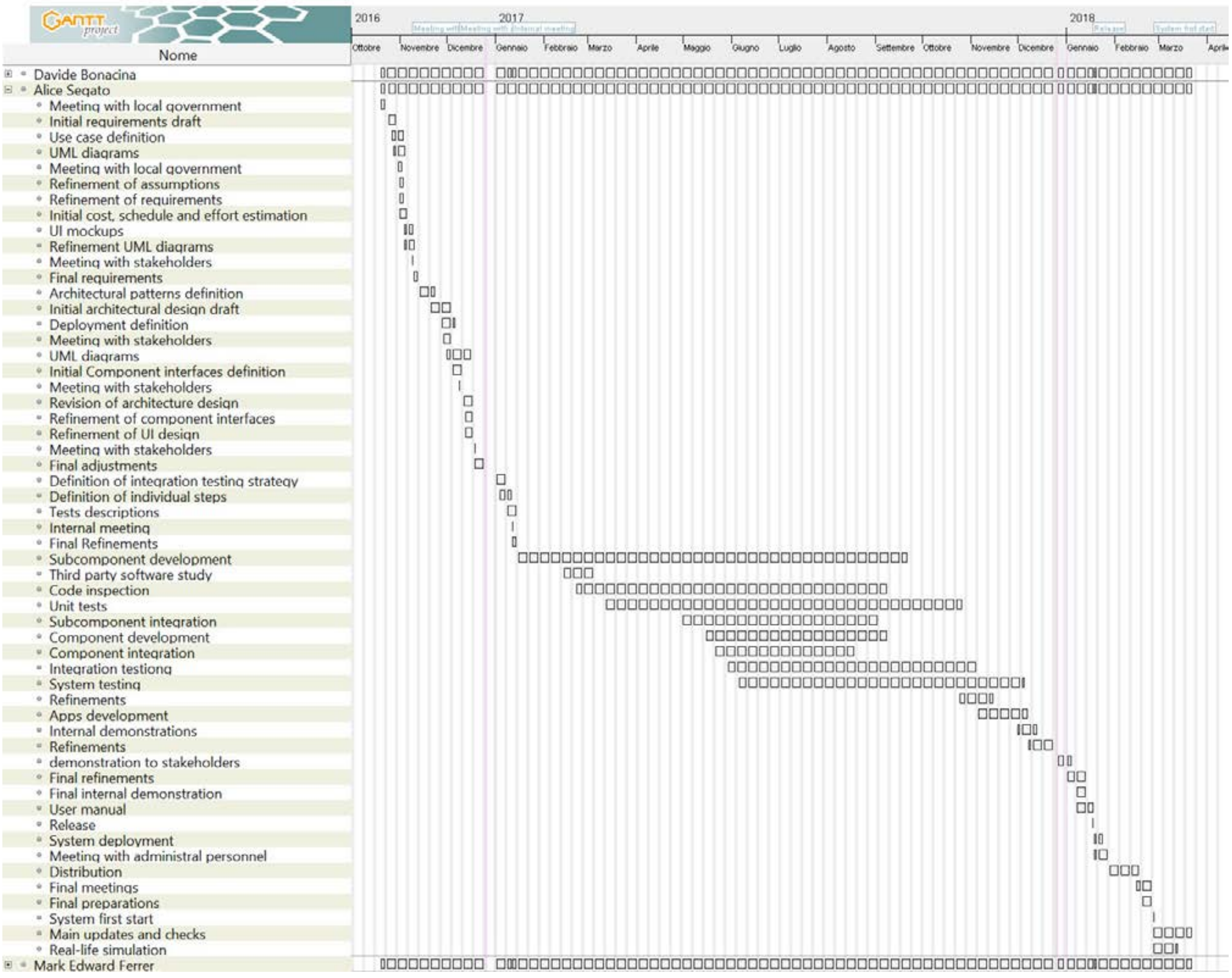


Nome

- Davide Bonacina
  - Meeting with local government
  - Domain properties definition
  - Initial requirements draft
  - Initial assumptions
  - Use case definition
  - UML diagrams
  - Meeting with local government
  - Refinement of assumptions
  - Refinement of requirements
  - Initial cost, schedule and effort estimation
  - Alloy modeling
  - Refinement UML diagrams
  - Meeting with stakeholders
  - Final requirements
  - Architectural patterns definition
  - Initial architectural design draft
  - Meeting with stakeholders
  - UML diagrams
  - Initial draft of algorithm design
  - Meeting with stakeholders
  - Revision of architecture design
  - Refinement of algorithm design
  - Meeting with stakeholders
  - Final adjustments
  - Definition of integration testing strategy
  - Definition of individual steps
  - Internal meeting
  - Final Refinements
  - Subcomponent development
  - Third party software study
  - Code inspection
  - Unit tests
  - Subcomponent integration
  - Component development
  - Component integration
  - Integration testiong
  - System testing
  - Refinements
  - Apps development
  - Internal demonstrations
  - Refinements
  - demonstration to stakeholders
  - Final refinements
  - Final internal demonstration
  - User manual
  - Release
  - System deployment
  - Meeting with administrial personnel
  - Distribution
  - Final meetings
  - Final preparations
  - System first start
  - Main updates and checks
  - Real-life simulation
- Alice Segato
- Mark Edward Ferrer

*Figure 7*

## 4.2 Alice Segato

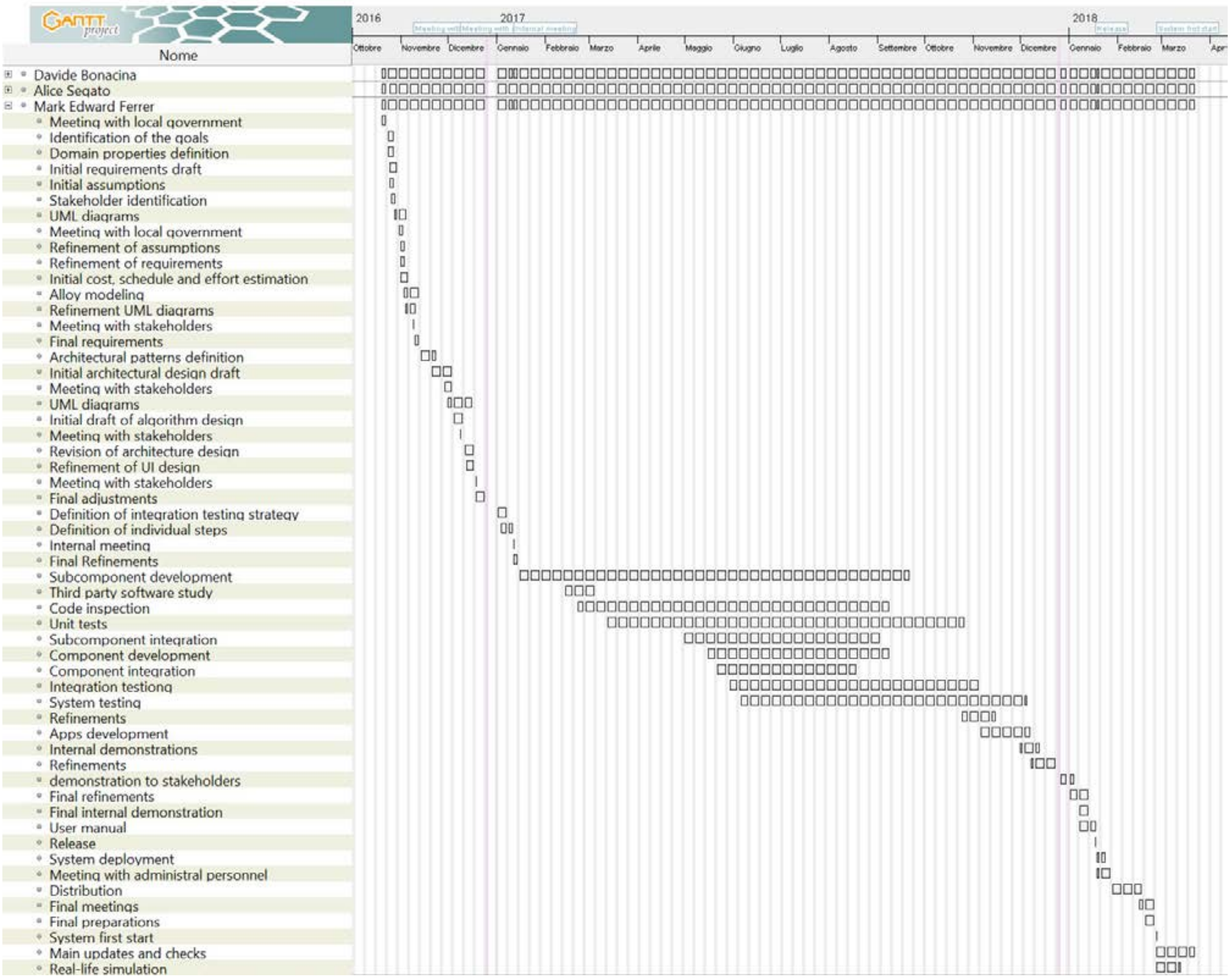


Nome

- Davide Bonacina
- Alice Segato
  - Meeting with local government
  - Initial requirements draft
  - Use case definition
  - UML diagrams
  - Meeting with local government
  - Refinement of assumptions
  - Refinement of requirements
  - Initial cost, schedule and effort estimation
  - UI mockups
  - Refinement UML diagrams
  - Meeting with stakeholders
  - Final requirements
  - Architectural patterns definition
  - Initial architectural design draft
  - Deployment definition
  - Meeting with stakeholders
  - UML diagrams
  - Initial Component interfaces definition
  - Meeting with stakeholders
  - Revision of architecture design
  - Refinement of component interfaces
  - Refinement of UI design
  - Meeting with stakeholders
  - Final adjustments
  - Definition of integration testing strategy
  - Definition of individual steps
  - Tests descriptions
  - Internal meeting
  - Final Refinements
  - Subcomponent development
  - Third party software study
  - Code inspection
  - Unit tests
  - Subcomponent integration
  - Component development
  - Component integration
  - Integration testiong
  - System testing
  - Refinements
  - Apps development
  - Internal demonstrations
  - Refinements
  - demonstration to stakeholders
  - Final refinements
  - Final internal demonstration
  - User manual
  - Release
  - System deployment
  - Meeting with administral personnel
  - Distribution
  - Final meetings
  - Final preparations
  - System first start
  - Main updates and checks
  - Real-life simulation
- Mark Edward Ferrer

*Figure 8*

## 4.3 Mark Edward Ferrer



Nome

- Davide Bonacina
- Alice Segato
- Mark Edward Ferrer
  - Meeting with local government
  - Identification of the goals
  - Domain properties definition
  - Initial requirements draft
  - Initial assumptions
  - Stakeholder identification
  - UML diagrams
  - Meeting with local government
  - Refinement of assumptions
  - Refinement of requirements
  - Initial cost, schedule and effort estimation
  - Alloy modeling
  - Refinement UML diagrams
  - Meeting with stakeholders
  - Final requirements
  - Architectural patterns definition
  - Initial architectural design draft
  - Meeting with stakeholders
  - UML diagrams
  - Initial draft of algorithm design
  - Meeting with stakeholders
  - Revision of architecture design
  - Refinement of UI design
  - Meeting with stakeholders
  - Final adjustments
  - Definition of integration testing strategy
  - Definition of individual steps
  - Internal meeting
  - Final Refinements
  - Subcomponent development
  - Third party software study
  - Code inspection
  - Unit tests
  - Subcomponent integration
  - Component development
  - Component integration
  - Integration testiong
  - System testing
  - Refinements
  - Apps development
  - Internal demonstrations
  - Refinements
  - demonstration to stakeholders
  - Final refinements
  - Final internal demonstration
  - User manual
  - Release
  - System deployment
  - Meeting with administral personnel
  - Distribution
  - Final meetings
  - Final preparations
  - System first start
  - Main updates and checks
  - Real-life simulation

*Figure 9*

# 5 Risk management

The project is associated with many project, technical and economic risks

## 5.1 PROJECT RISKS

| Risk | Probability | Effect |
|---|---|---|
| Delays | High | Moderate |
| Lack of communication | Low | Moderate |
| Lack of experience | Certain | Moderate |
| Requirements change | Very low | Moderate |
| Team member quit/ill | Moderate | High |

### Delays over the expected deadlines:

The project could require more time than expected. If this happens, we may release a first, incomplete but working version (e.g. without a…) and build the less essential features later.

### Lack of communication among team members:

The team often works remotely and this can lead to misunderstandings in fundamental decisions, to conflicts over the division of the work among team members and to conflicts in code versioning. Those difficulties can be overcome by explicitly defining (e.g. for every delivery we previous discuss the tasks of every member of the group according with our capacity in various works and obviously talking to each other when some member of the group find some difficulties) the responsibilities of each team member, and by writing clear and complete specification and design documents.

### Lack of experience in programming with the specific frameworks:

The team has a little experience with Java EE programming but not enough to manage every single aspect of the development platform. This lack of experience leads to delays.
One way to mitigate this risk is to instruct the team on how to use the development platform, this means that it has to be outlined as soon as possible, possibly during the requirements analysis, in order to be prepared for the implementation phase.

### Requirements change:

The requirements may change during the development in unexpected way. This risk can't be prevented, but can be mitigated by writing reusable and extensible software.

### Team member quit/ill:

One or more of the team members could quit for any problem or he can be ill in a critical moment of the development. This problem leads to considerable shift in the scheduled tasks, especially if this risk happens in the last phases of the development.
This risk can be prevented dividing the tasks between a larger number of people: this can help to lower the difficulty of the sub tasks and prevents to big shifts in case of a quit. Obviously this can be done in a large team, so another way to mitigate the problem is to hire more developers.

## 5.2 TECHNICAL RISKS

| Risk | Probability | Effect |
|---|---|---|
| Integration testing failure | Low | High |
| Downtime | Moderate | High |
| Scalability issues | Low | Moderate |
| Spaghetti code | Low | Moderate |
| Deployment difficulties | Moderate | Moderate |
| Data loss | Low | Catastrophic |
| Data leaks | Moderate | Catastrophic |
| Incompatibility | Moderate | Moderate |

### Integration testing failure:

After the implementation of some components, they may not pass the integration testing phase: this can require to rewrite large pieces of software. This risk can be mitigated by defining precisely the interfaces between components and subsystems, and by doing integration tests.

### Downtime:

The system could go down for excessive load, software bugs, hardware failures or power outages. This risk can be mitigated by building redundant systems and placing them in geographically separate data centers and by performing testing at all levels.

### Scalability issues:

The system could not scale with a large number of users, requiring a major design rework. A possible plan is to use a cloud infrastructure from a third-party provider to host our system.

### Spaghetti code:

With the growth of the project, the code may become overloaded, badly structured and unreadable. This risk can be mitigated by writing a good Design Document before starting to code, and by performing code inspection periodically.

### Deployment difficulties:

If cities already have a Car Sharing system, it could be difficult to deploy our system by keeping and migrating the old data. This problem can be solved by hiring professional system integrators, but of course this would increase the costs.

### Data loss:

Data can be lost because of hardware failures, misconfigured software or deliberate attacks. This problem can be prevented by enforcing a reliable backup plan. Backups should be kept in a separate place from the system, and offline.

### Data leaks:

Misconfiguration, software bugs and deliberate attacks can expose the users’ personal data. This risk must be prevented by adopting industry security standards, by encrypting communications and by doing regular penetration testing.

**Incompatibility:**
the system works on and with different hardware that can be incompatible in some ways (e.g. the car computer is implemented with a different API as expected or the physical server cannot host the system for architectural problems). This risk can be mitigated by checking the chosen hardware before the implementation phase to ensure that the finished product will fit into the hardware.

## 5.3 ECONOMICAL RISKS

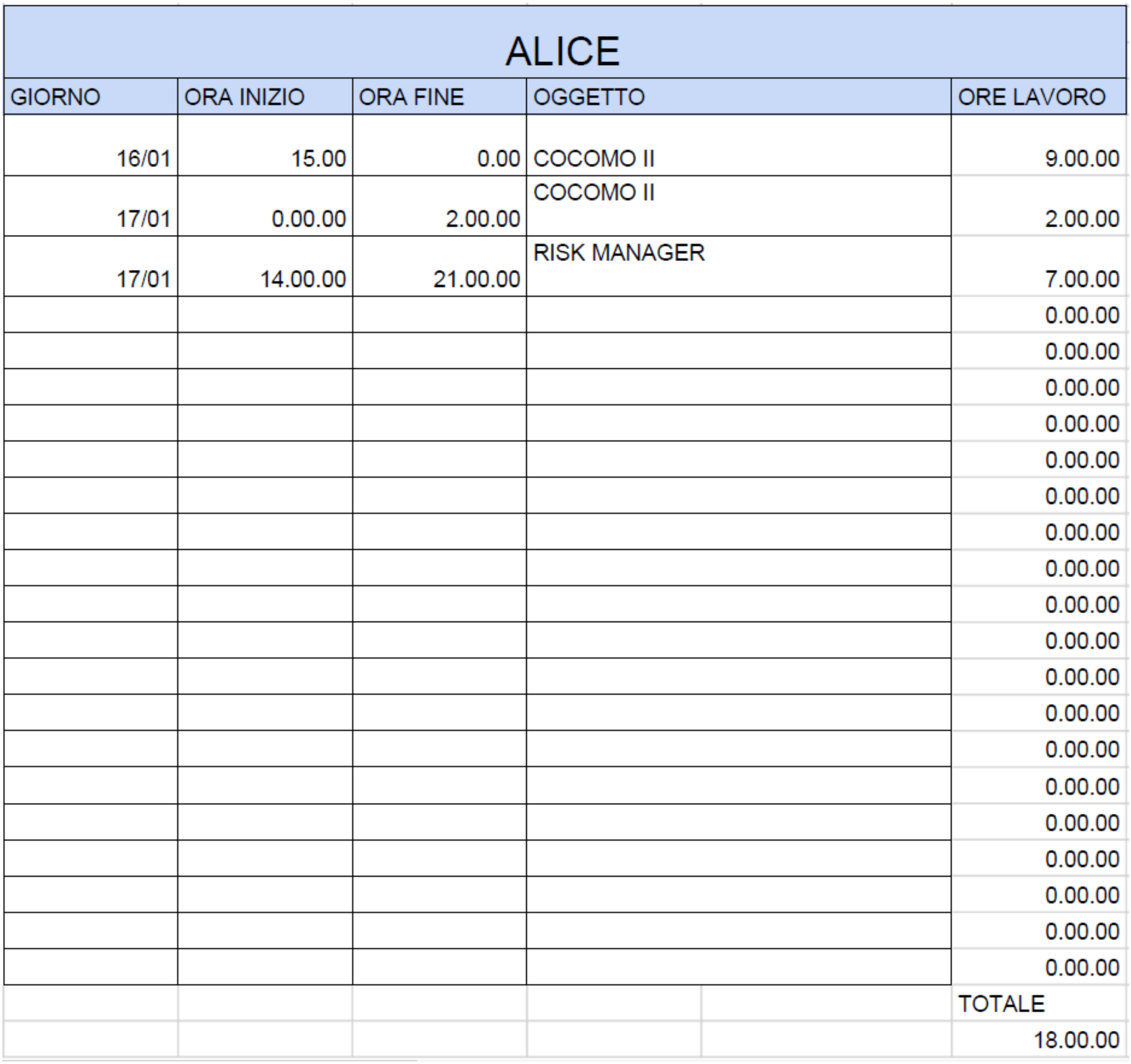
| Risk | Probability | Effect |
|---|---|---|
| Bankruptcy | Moderate | Catastrophic |
| Regulation change | Low | Severe |
| Competitors | Moderate | Severe |

**Bankruptcy:**
The income from the sales of the software may not be enough to sustain the development, maintenance and deployment of the software system. A good feasibility study helps to avoid this situation asking maybe also some incentive by the city for using green car that take care about the environment.

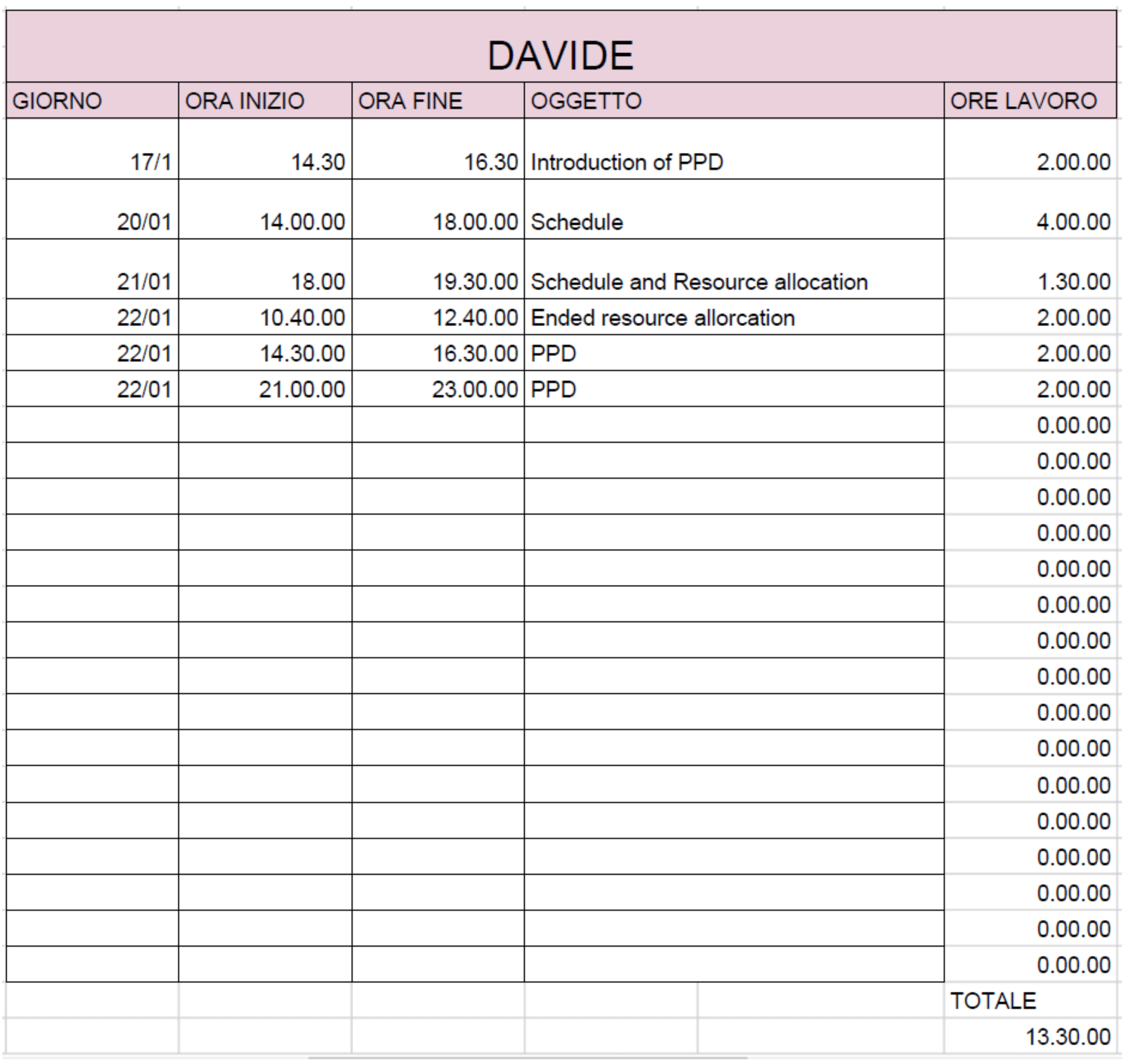
**Regulation change:**
Local and State regulators can change the car sharing regulation at any time, and this could have an unpredictable impact on the usage and the market penetration of our software. This risk can be partially avoided by a good feasibility study. This issue could be mitigated by doing lobbying on the political parties of the affected countries (wherever this is legal). On the other part as in PowerEnJoy case we can be helped by local and State regulator as doing more green car that take care of the environment and in this way people can have more benefit using our cars.

**Competitors:**
Other companies like Enjoy or car2GO as can build equivalent or better products at a lower price and put PowerEnJoy out of the market. The competitiveness of our product must be continuously enhanced by introducing innovative features or introduce discounts as the stakeholders yet asked to implement.

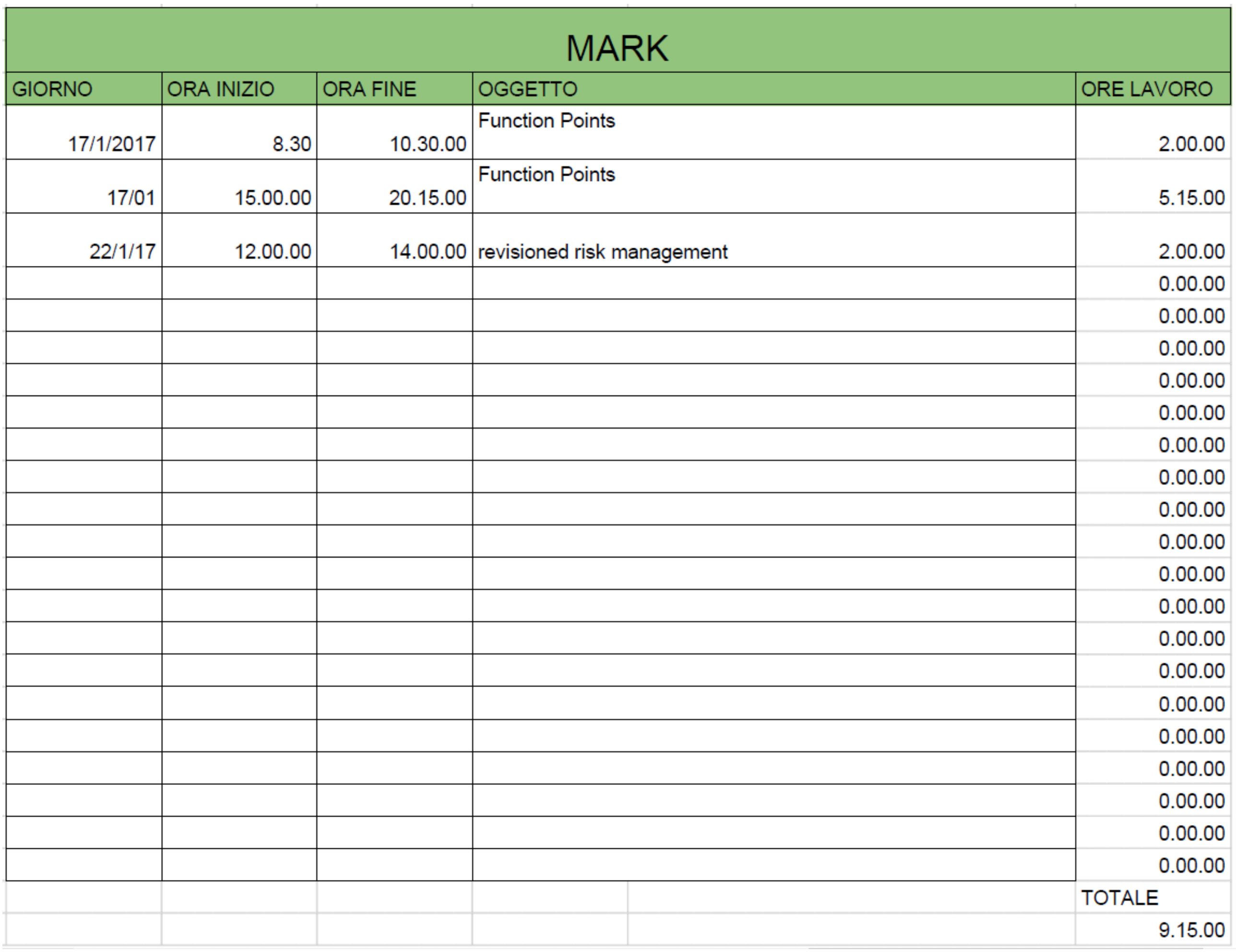
# 6 Hours of work



| ALICE | | | | |
|---|---|---|---|---|
| GIORNO | ORA INIZIO | ORA FINE | OGGETTO | ORE LAVORO |
| 16/01 | 15.00 | 0.00 | COCOMO II | 9.00.00 |
| 17/01 | 0.00.00 | 2.00.00 | COCOMO II | 2.00.00 |
| 17/01 | 14.00.00 | 21.00.00 | RISK MANAGER | 7.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | TOTALE |
| | | | | 18.00.00 |



| DAVIDE | | | | |
|---|---|---|---|---|
| GIORNO | ORA INIZIO | ORA FINE | OGGETTO | ORE LAVORO |
| 17/1 | 14.30 | 16.30 | Introduction of PPD | 2.00.00 |
| 20/01 | 14.00.00 | 18.00.00 | Schedule | 4.00.00 |
| 21/01 | 18.00 | 19.30.00 | Schedule and Resource allocation | 1.30.00 |
| 22/01 | 10.40.00 | 12.40.00 | Ended resource allorcation | 2.00.00 |
| 22/01 | 14.30.00 | 16.30.00 | PPD | 2.00.00 |
| 22/01 | 21.00.00 | 23.00.00 | PPD | 2.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | TOTALE |
| | | | | 13.30.00 |



| MARK | | | | |
|---|---|---|---|---|
| GIORNO | ORA INIZIO | ORA FINE | OGGETTO | ORE LAVORO |
| 17/1/2017 | 8.30 | 10.30.00 | Function Points | 2.00.00 |
| 17/01 | 15.00.00 | 20.15.00 | Function Points | 5.15.00 |
| 22/1/17 | 12.00.00 | 14.00.00 | revisioned risk management | 2.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | 0.00.00 |
| | | | | TOTALE |
| | | | | 9.15.00 |