POLITECNICO DI MILANO

POWERENJOY

SOFTWARE ENGINEERING 2

# Project Planning

*Authors:*
Giancarlo COLACI
Giulio DE PASQUALE
Francesco RINALDI

*Supervisor:*
Elisabetta DE NITTO

January 22, 2017

# Contents

# 1 Introduction

## 1.1 Revision History

| Version | Date | Author(s) | Summary |
|---|---|---|---|
| 1.0 | 19/01/2017 | Giancarlo Colaci, Giulio De Pasquale, Francesco Rinaldi | Initial Release |

## 1.2 Reference Documents

1. QSM - Function Point Languages Table[1]
2. COCOMO II - Model Definition Manual [2]
3. CCM Maturity Questionnaire [3]

[1] http://www.qsm.com/resources/function-point-languages-table
[2] http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf
[3] http://resources.sei.cmu.edu/asset_files/SpecialReport/1994_003_001_16265.pdf

# 2 Function Point

A function point is a "unit of measurement" representing how many functionalities the information system provides to the user. Function points are used to calculate a functional size measurement (FSM) of software; usually the cost of a single unit is calculated from past projects.

## 2.1 Overview

Several aspects are considered for the estimation, as prescribed by the specifications:

- **Internal Logic Files**: homogeneous set of data handled by the application being developed;
- **External Interface Files**: homogeneous set of data managed by the application but created elsewhere;
- **External Input**: basic operation involving externally provided data as input;
- **External Inquiry**: basic operation involving both input and output, mainly for retrieving information from the system;
- **External Output**: basic operation which provides data to the external environment.

The **counting weight** (Low, Avg. or High) has been defined according to the parameters specified in Tables 1-3. Finally, the **function points** has been calculated for each section according to Table 4 in order to get the estimated SLOC size, as described in Section 2.7.

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

Table 1: Internal Logical Files and External Interface Files

| **File Types** | **Data Elements** | | |
|---|---|---|---|
| | **1-5** | **6-19** | **20+** |
| **0-1** | Low | Low | Avg |
| **2-3** | Low | Avg | High |
| **4+** | Avg | High | High |

Table 2: External Output and External Inquiry

| File Types | Data Elements | | |
|---|---|---|---|
| | **1-4** | **5-15** | **16+** |
| **0-1** | Low | Low | Avg |
| **2-3** | Low | Avg | High |
| **4+** | Avg | High | High |

Table 3: External Input

| Function Types | Complexity Weight | | |
|---|---|---|---|
| | **Low** | **Average** | **High** |
| **Internal Logical 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 |

Table 4: Complexity Weights

## 2.2 Internal Logic Files

The system must handle data about the following classes:

| File | Records | Data Elements | Counting Weight | FPs |
|---|---|---|---|---|
| **Car** | 6+ | 51+ | High | 15 |
| **RMSS** | 6+ | 51+ | High | 15 |
| **User** | 6+ | 51+ | High | 15 |
| **Location** | 2-5 | 51+ | High | 15 |
| **Event** | 2-5 | 51+ | High | 15 |
| ***Grand Total*** | | | | **75** |

## 2.3 External Interface Files

The system must store the following data from external environment:

| File | Records | Data Elements | Counting Weight | FPs |
|---|---|---|---|---|
| **Maps** | 2-5 | 51+ | High | 10 |
| ***Grand Total*** | | | | **10** |

## 2.4 External Input

The system must guarantee the following oprations using input from the external environment:

| File | Classes Involved | Data Elements | Counting Weight | FPs |
|---|---|---|---|---|
| **Login / Sign up / Logout** | 1 | 16+ | Average | 3x4 |
| **Edit / Delete Profile** | 1 | 16+ | Average | 2x4 |
| **Add / Delete Reservation** | 1 | 16+ | Average | 2x4 |
| **Add / Delete Rent** | 1 | 16+ | Average | 2x4 |
| **Set Car Status** | 1 | 16+ | Average | 4 |
| ***Grand Total*** | | | | **40** |

## 2.5 External Inquiry

The system must respond to the following requests:

| File | Classes Involved | Data Elements | Counting Weight | FPs |
|---|---|---|---|---|
| **Get reservation history** | 2 | 20+ | High | 6 |
| **Get rent history** | 2 | 20+ | High | 6 |
| **Get available cars in radius** | 2 | 20+ | High | 6 |
| **Check reservation status** | 1 | 20+ | Average | 4 |
| **Check rent status** | 1 | 20+ | Average | 4 |
| **Calculate rent fee** | 1 | 20+ | Average | 4 |
| ***Grand Total*** | | | | **30** |

## 2.6 External Output

The system must produce data to the external environment through the following operations:

| File | Classes Involved | Data Elements | Counting Weight | FPs |
|---|---|---|---|---|
| **Confirmation emails** | 1 | 20+ | Average | 5 |
| **Notification to users** | 2 | 20+ | High | 7 |
| **Payment requests** | 2 | 20+ | High | 7 |
| **Maintainance requests** | 2 | 20+ | High | 7 |
| ***Grand Total*** | | | | **26** |

## 2.7 Results

Following the tables described in [1], for **J2EE**, we have that

$$\frac{SLOC}{FPs} = 46$$

Using the tables listed before we get:

$$SLOC = 46 \cdot 181 = 8326$$

# 3 COCOMO®: effort & cost estimation

## 3.1 Overview

The COCOMO R II Cost Estimation Model is a complex estimation technique used by thousands of software engineers all over the world. It is used to estimate the effort cost of a software engineering project. The core of COCOMO R II is the use of the Effort Equation to estimate the number of Person/Month required to develop a complex project.

## 3.2 Scale Drivers

In this section we will talk about COCOMO R II Scale Drivers. They are a significant source of exponential variation on a project effort. Each driver has a range of rating levels, from Very Low to Extra High, each with its own weight, that is called "Scale Factor" (SF).

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

Table 5: Scale Factor Values for COCOMO R II Models

#### Precedentedness

**PREC** This driver reflects the previous experience that developers have in the development of large scale projects. Actually, this is our first experience, so we think the best value for our team is Low.

#### Development flexibility

**FLEX** This driver reflects the degree of flexibility in the development process with respect to the external specication and requirements. Since on side there are very strict requirements on the functionalities but on the other no specific technology was specified to be used, this value will be Low.

#### Risk resolution

**RESL** It reflects the extension of risk analysis. A Very Low value means we have done a poor analysis, Very High means a complete risk analysis. We choose Very High because we did a detailed analysis (Chapter 6).

#### Team cohesion

**TEAM 5** This value is an indicator of how well the development team know each other. In this case we are a very cooperative team, so we choose a Very High value.

#### Process maturity

**PMAT** This parameter reflects the process maturity of the organizazion. In particular, this parameter has been choosen according to a weighted average of "Yes" answers to CMM Maturity Questionnaire. In our case we have chosen High (CMM Level 3).

**Results** The results of our evaluation is summed up in the following:

| Code | Name | Factor | Value |
|---|---|---|---|
| PREC | *Precedentedness* | Low | 4.96 |
| FLEX | *Development exibility* | Low | 4.05 |
| RESL | *Risk resolution* | Very High | 1.41 |
| TEAM | *Team cohesion* | Very High | 1.10 |
| PMAT | *Process maturity* | Level 3 | 3.12 |
| **Total** | $E = 0.91 + 0.01 \times \sum_i SF_i = 0.91 + 0.01 \times 14.64$ | | 1.0564 |

## 3.3 Cost Drivers

These are the effort multipliers used in COCOMO R II model to adjust the nominal effort.

### Required Software Reliability

**RELY** This is the measure of software reliability. Since the system represents the only way to reserve PowerEnjoy electric cars, a downtime could lead to important financial losses. For this reason we choose High value.

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

Table 6: RELY Cost Drivers

### Database Size

**DATA** This values tries to estimate effects that large databases could have in our application. We have not a precise answer, because we can just give an estimation of the DB size, around the 3GB. Since it is distributed over 8.000 - 9.000 SLOC (precisely 8326, as we see at section 2.7 of this document), the ratio D/P (measured as testing DB bytes / program SLOC) is between 334 and 375, resulting in the DATA cost driver being High.

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

Table 7: DATA Cost Drivers

### Product Complexity

**CPLX** According to the table II-15: Module Complexity Ratings versus Type of Module of the COCOMO II Model Definition Manual, our software could be marked as High.

| 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 |

Table 8: CPLX Cost Drivers

#### Required Reusability

**RUSE** Reusability is a cardinal principle for every kind of projects: we designed several parts of the system-to-be in a way they would result reusable. For this reason, we decided to set Nominal this value.

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

Table 9: RUSE Cost Drivers

#### Documentation match to life-cycle needs

**DOCU** This parameter is a measure of the he suitability of the project's documentation to its life-cycle needs. In our case, every need of the product life-cycle is already foreseen in our documentation, so we set the DOCU cost driver to Nominal.

| Rating level | Very low | Low | Nominal | High | Very High | Extra High |
|---|---|---|---|---|---|---|
| **DOCU Descriptors** | Many life-cycle needs uncovered | Some life-cycle needs uncovered | Right-sized to life-cycle needs | Excessive for life-cycle needs | Very excessive for life-cycle needs | Across multiple product lines |
| **Effort multipliers** | 0.81 | 0.91 | 1.00 | 1.11 | 1.23 | 1.24 |

Table 10: DOCU Cost Drivers

#### Execution Time Constraint

**TIME** This is a measure of the execution time constraint imposed upon a software system. The rating is expressed in terms of the percentage of available execution time expected to be used by the system or subsystem consuming the execution time resource. We set this value to Nominal.

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

Table 11: TIME Cost Drivers

**Main Storage Constraint**

**STOR** This rating represents the degree of main storage constraint imposed on a software system or subsystem. It is relevant to set a value also for this cost driver because many applications continue to expand to consume other resources. We set this value to Nominal.

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

Table 12: STOR Cost Drivers

**Platform Volatility**

**PVOL** Here with the term “platform” we mean the complex of hardware and software (OS, DBMS, etc..). We do not expect to change our platforms very often. However, the client applications may require at least a major release once every six months to be aligned with the development cycle of the main mobile operating systems. For this reason, this parameter is set to Nominal.

| Rating level | Very low | Low | Nominal | High | Very High | Extra High |
|---|---|---|---|---|---|---|
| **PVOL Descriptors** | | Major change every 12 months, minor change every 1 month | Major change every 6 months, minor change every 2 weeks | Major change every 2 months, minor change every 1 week | Major change every 2 weeks, minor change every 2 days | |
| **Effort multipliers** | n/a | 0.87 | 1.00 | 1.15 | 1.30 | n/a |

Table 13: PVOL Cost Drivers

#### Analyst Capability

**ACAP** Analysts are personnel that work on requirements, high level design and detailed 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 driver should be set to High since we spent a lot of time in analysing the problem requirements.

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

Table 14: ACAP Cost Drivers

#### Programmer Capability

**PCAP** This driver should evaluate 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. The experience of the programmer should not be considered here. Our cooperation is quite good, so we set it as High.

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

Table 15: PCAP Cost Drivers

#### Application Experience

**AEXP** This rating is dependent on the level of applications experience of the project team developing the software system or subsystem. Our experience in this field is very low. So we think that a good estimate will happen if we set this value to Low.

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

Table 16: AEXP Cost Drivers

#### Platform Experience

**PEXP** Our average knowledge about platforms as databases, UI and client/server architecture is low. For this reason, we set this value as Low.

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

Table 17: PEXP Cost Drivers

#### Language and Tool Experience

**LTEX** This is a measure of the level of programming language and software tool experience of the project team developing the software system or subsystem. Our experience is less than 3 years, so this value will be set to Nominal.

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

Table 18: LTEX Cost Drivers

#### Personnel Continuity

**PCON** The rating scale for PCON is in terms of the project's annual personnel turnover: from 3%, very high, to 48%, very low. We estimated a High personnel continuity.

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

Table 19: PCON Cost Drivers

#### Use of Software Tools

**TOOL** The tool rating ranges from simple edit and code, very low, to integrated lifecycle management tools, very high. We set this value to Nominal.

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

Table 20: TOOL Cost Drivers

**Multisite Development**

**SITE** Given the increasing frequency of multisite developments, the SITE cost driver has been added in COCOMO II. Determining its cost driver rating involves the assessment and averaging of two factors: site collocation (from fully collocated to international distribution) and communication support (from surface mail and some phone access to full interactive multimedia). We worked in the same city, communicating by phone, chat and emails. For this reason we set this value to High.

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

Table 21: SITE Cost Drivers

**Required Development Schedule**

**SCED** 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. Accelerated schedules tend to produce more effort in the later phases of development because more issues are left to be determined due to lack of time to resolve them earlier. A schedule compress of 74% is rated very low. A stretch-out of a schedule produces more effort in the earlier phases of development where there is more time for thorough planning, specification and validation. A stretch-out of 160% is rated very high. We put a constant effort into this project, so one hundred percent is a good percentage to describe our work. For this reason we choose Nominal here.

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

Table 22: SCED Cost Drivers

**Results** Now we can compute the product of all Cost Drivers.

| Code | Name | Factor | Value |
|---|---|---|---|
| RELY | *Required Software Reliability* | High | 1.10 |
| DATA | *Database Size* | High | 1.14 |
| CPLX | *Product Complexity* | High | 1.17 |
| RUSE | *Required Reusability* | Nominal | 1.00 |
| DOCU | *Documentation match to life-cycle needs* | Nominal | 1.00 |
| TIME | *Execution Time Constraint* | Nominal | 1.00 |
| STOR | *Main Storage Constraint* | Nominal | 1.00 |
| PVOL | *Platform Volatility* | Nominal | 1.00 |
| ACAP | *Analyst Capability* | High | 0.85 |
| PCAP | *Programmer Capability* | High | 0.88 |
| AEXP | *Application Experience* | Low | 1.10 |
| PEXP | *Platform Experience* | Low | 1.09 |
| LTEX | *Language and Tool Experience* | Nominal | 1.00 |
| PCON | *Personnel Continuity* | High | 0.90 |
| TOOL | *Use of Software Tools* | Nominal | 1.00 |
| SITE | *Multisite Development* | High | 0.93 |
| SCED | *Required Development Schedule* | Nominal | 1.00 |
| **Total** | $EAF = \prod_i C_i =$ product of all cost drivers = | | 1.1014 |

### 3.4 Effort Equation

Now, having both cost drivers product and scale drivers factors, we can compute the Effort, in Person-Month, with the following equation:

$$Effort = A \times EAF \times KSLOC^{E}$$

where:

- A = 2.94 (for COCOMO II)
- EAF = the product of all cost drivers; we calculated it above, and its value is 1.1014
- E = the exponent derived from the scale drivers; we calculated it above, through this formula:

$$E = B + 0.01 \times \sum_{i} SF_i = 0.91 + 0.01 \times 14.64 = 1.0564$$

By substituting these values into the formula of the Effort written above, we obtain:

$$Effort = A \times EAF \times KSLOC^{E} =$$

$$= 2.94 \times 1.1014 \times 8326^{1.0564} =$$

$$= 44.858PM \approx 45PM$$

### 3.5 Schedule Estimation

Regarding the final schedule, we will use this formula to obtain the Duration:

$$Duration = 3.67 \times Effort^{F}$$

where:

- Effort, in Person-Month, obtained above; its value is 45 PM
- $F = 0.28 + 0.2 \times (E - B) = 0.28 + 0.2 \times (1.0564 - 0.91) = 0.28 + 0.2 \times 0.1464 = 0.30928$

By substituting these values into the formula of the Duration written above, we obtain:

$$Duration = 3.67 \times Effort^{F} =$$

$$= 3.67 \times 45^{0.30928} = 11.911642\, months$$

# 4 Tasks scheduling

The main assignments of our project are:

1. Creation of the Requirement Analysis and Specification Document (**RASD**).
2. Creation of the Design Document (**DD**).
3. Creation of the Integration Testing Plan Document (**ITPD**).
4. Creation of the Project Plan, this document.
5. Creation of a set of slides to present our work to the client.
6. Development of the system and the preparation of the unit tests.
7. Running of integration testing on the application.

The first four tasks were already completed within the given submission deadlines.

Starting from step 5 onward, according to our COCOMO® estimation [4], we expect to deliver a working implementation within **12** months which corresponds to **November 2017.** The development started just after the submission of the Design Document and continued simultaneously with the rest of the tasks; the integration testing will take place in the last month.

In the meantime, tests will be run all along the entire development process to verify the proper functioning of every added feature.

In Figure 1 you can find the dependency graph of every task. Also the Gantt chart for the project is provided in Figure 2.

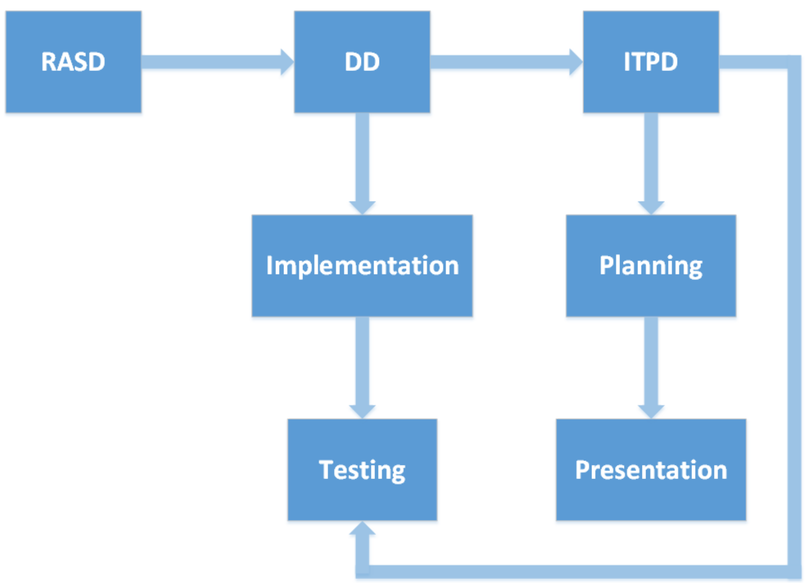


Figure 1: Dependency graph

[4]see Section 3 for more details

| ID | Task | Start Date | End Date | Duration | Assigned To |
|---|---|---|---|---|---|
| 1 | **RASD** | **16/10/2016** | **13/11/2016** | **29g** | |
| 2 | Introduction | 16/10/2016 | 23/10/2016 | 8g | C, DP, R |
| 3 | Description | 24/10/2016 | 30/10/2016 | 7g | C |
| 4 | Specific Requirements | 24/10/2016 | 30/10/2016 | 7g | DP |
| 5 | Scenarios | 24/10/2016 | 30/10/2016 | 7g | R |
| 6 | Models | 31/10/2016 | 06/11/2016 | 7g | C, DP, R |
| 7 | Alloy | 07/11/2016 | 13/11/2016 | 7g | C, DP, R |
| 8 | Revision | 07/11/2016 | 13/11/2016 | 7g | C, DP, R |
| 9 | **Design Document** | **14/11/2016** | **11/12/2016** | **28g** | |
| 10 | Introduction | 14/11/2016 | 20/11/2016 | 7g | DP |
| 11 | Architectural Design | 14/11/2016 | 27/11/2016 | 14g | C, DP, R |
| 12 | Algorithms | 21/11/2016 | 04/12/2016 | 14g | DP, R |
| 13 | User Interface | 28/11/2016 | 04/12/2016 | 7g | C, R |
| 14 | Revision | 05/12/2016 | 11/12/2016 | 7g | C, DP, R |
| 15 | **ITPD** | **12/12/2016** | **15/01/2017** | **35g** | |
| 16 | Introduction | 12/12/2016 | 18/12/2016 | 7g | C |
| 17 | Integration Strategy | 12/12/2016 | 18/12/2016 | 7g | DP, R |
| 18 | Individual Steps | 19/12/2016 | 25/12/2016 | 7g | C |
| 19 | Test / Tools Descriptions | 19/12/2016 | 25/12/2016 | 7g | DP |
| 20 | Program Stubs and Test Data Required | 19/12/2016 | 25/12/2016 | 7g | R |
| 21 | Revision | 09/01/2017 | 15/01/2017 | 7g | C, DP, R |
| 22 | **Planning** | **16/01/2017** | **22/01/2017** | **7g** | |
| 23 | COCOMO | 16/01/2017 | 22/01/2017 | 7g | C, R |
| 24 | Risks | 16/01/2017 | 22/01/2017 | 7g | C |
| 25 | Resource Allocation | 16/01/2017 | 22/01/2017 | 7g | DP |
| 26 | Tasks scheduling | 16/01/2017 | 22/01/2017 | 7g | DP |
| 27 | Function Points | 16/01/2017 | 22/01/2017 | 7g | R |
| 28 | **Implementation** | 27/11/2016 | 27/11/2017 | 366g | |

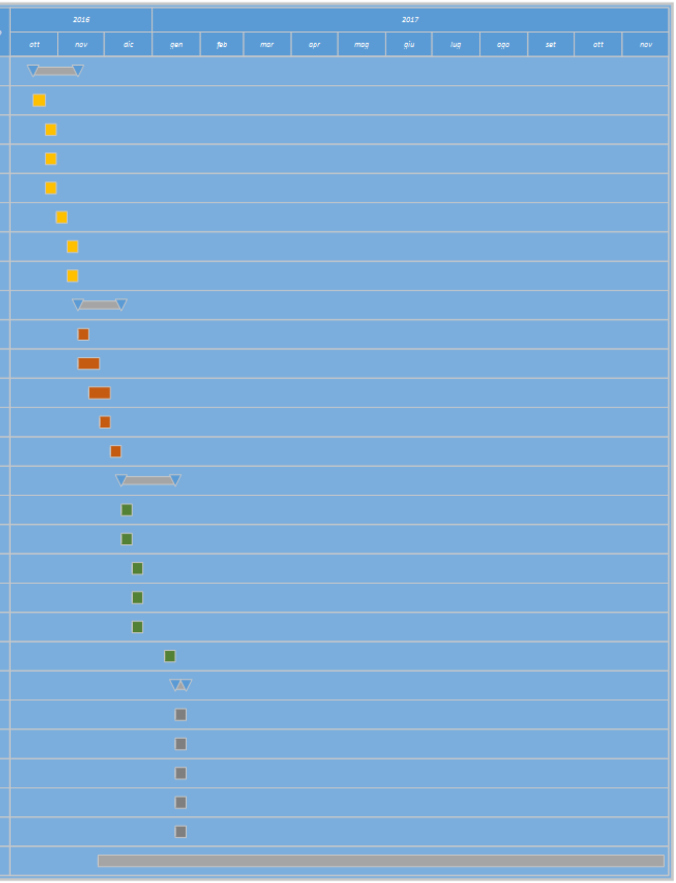


Figure 2: GANTT graph

# 5 Resource Allocation

This section shows how we distributed our resources for the project.

Every assignment has been divided into **several sub-assignments**, each of them delegated to a team member. We decided to cross-check every assignment in order to minimize any misunderstanding about the ideas involved in the project: this may have increased the time spent on each phase but no action was taken with any doubt by any of the members. Each task has been revisioned by the whole team once it was completed, usually one week to 3 days before the submission deadline.

Once the Design Document has been completed, each team member has focused on the **implementation and consequent integration testing**: when possible, this was done simultaneously by every component of the team. Whenever a feature is completed, the related testing has to be carried on by another team member to make the unit test more accurate.

The following tables explain further the work division throughout the whole project.

| **Member** | $1^{st}$ week | $2^{nd}$ week | $3^{rd}$ week | $4^{th}$ week |
|---|---|---|---|---|
| **Colaci** | Introduction | Description | Models, Use Case Diagram | Alloy, Revision |
| **De Pasquale** | Introduction | Specific Requirements | Use Case Diagram, Class Diagram | Alloy, Revision |
| **Rinaldi** | Introduction | Scenarios | Use Case Diagram, Class Diagram | Alloy, Revision |

Table 23: **RASD:** Resource Allocation from *16/10/16* to *13/11/16*

| **Member** | $1^{st}$ week | $2^{nd}$ week | $3^{rd}$ week | $4^{th}$ week |
|---|---|---|---|---|
| **Colaci** | Architectural Design | Architectural Design | User Interface | Revision |
| **De Pasquale** | Introduction | Architectural Design | Algorithms | Revision |
| **Rinaldi** | Architectural Design | Algorithms | User Interface | Revision |

Table 24: **Design Document:** Resource Allocation from *14/11/16* to *11/12/16*

| **Member** | $1^{st}$ week | $2^{nd}$ week | $3^{rd}$ and $4^{th}$ week | $5^{th}$ week |
|---|---|---|---|---|
| **Colaci** | Introduction | Individual Steps | | Revision |
| **De Pasquale** | Integration Strategy | Test / Tools Descriptions | | Revision |
| **Rinaldi** | Integration Strategy | Program Stubs and Test Data Required | | Revision |

Table 25: **ITPD:** Resource Allocation from *12/12/16* to *15/01/17*

| Member | $1^{st}$ week |
|---|---|
| **Colaci** | COCOMO, Risks |
| **De Pasquale** | Tasks scheduling, Resource Allocation |
| **Rinaldi** | Function Points, COCOMO |

Table 26: **Planning:** Resource Allocation from *16/01/17* to *22/01/17*

# 6 Risks

Risks have always to be considered in any long term project planning due to their uncertain nature. The whole development could fail suddenly due to external actions, economical situations or architectural changes; this is the reason why they are here analyzed. Three main risk categories will be later described:

- **Business risks**, involving the company developing the software.
- **Project risks**, involving the project plan.
- **Technical risks**, involving the project implementation.

## 6.1 Business Risks

| Risk | Probability | Damage | Possible solution |
|---|---|---|---|
| PowerEnjoy can potentially violate some future laws regarding car sharing. | Low | Critical | Frequent checks has to be conducted in order to avoid possible lawsuits. In case of sudden and critical changes, the team has to adapt to the new regulations as fast as possible. |
| A company may acquire our firm. | Medium | Marginal | No preventive solutions are available. This is not strictly bad news. |
| The company may find itself in serious financial trouble such as bankruptcy. | Low | Critical | An in depth analysis of the RASD along with a feasibility research has to highlight the inability to start a new project. |
| The infrastructure along with every device (mobile phones, PCs, servers) need to be repaired, purchased or configured. This is going to increase costs, that may be not sustainable if the company is too small. | High | Critical | Testing tools and software suites have to be clearly defined in order to avoid worthless spendings. |

## 6.2 Project Risks

| Risk | Probability | Damage | Possible solution |
|---|---|---|---|
| A sudden illness or termination of a team member may bring several repercussion due to the small size of the group. | Low | Critical | The remaining team members must be cooperate effectively and have to be able to continue the development. |
| No schedules or estimations have been made before this project. Lack of experience in this area can lead to major errors in evaluating development time. | High | Critical | Studying previous works and projects on a similar subject can be very helpful. |
| A requirements growth can lead to rush meeting deadlines, severely decreasing the overall quality. | Medium | Critical | The team has to distinguish over-engineering from actual requirements; furthermore, planning the first stages with a broader can be very helpful. |
| Collaboration issues can sometimes be crucial, especially when dealing with task divisions. | Medium | Medium | Periodic meetings help the team to be constantly organized and not overwhelmed by tasks. |
| The project may be delayed to multiple overlapping tasks | High | Critical | A good organization among the team components is fundamental. This leads to a better teamwork which allows to maximize the throughput. |

## 6.3 Technical Risks

| Risk | Probability | Damage | Possible solution |
|---|---|---|---|
| The testing phase may be harder than expected and / or highlight bugs that are hard to solve. | Medium | Critical | Every component has to be tested as soon as possible in order to solve critical bugs and integration testing has to be executed as defined in the ITPD. A requirements check has to be done periodically. |
| Lack of experience in the new environment (e.g JavaEE) may slow down the development or other experienced team members. | High | Critical | This has to be taken into account in the first stages of planning and put into the project scheduling. |
| Security bugs may be suddenly discovered if the application is not well designed. | Medium | Critical | Computer security guidelines have to be followed in order to minimize the number of incidents; when dealing with user inputs, each field has to be sanitized. |
| A significant downtime can critically damage the whole project if the servers are unreliable or more than the expected number of users use the service. | Medium | Critical | The architecture has to adopt a fully scalable design, both software and hardware side. |

# 7 Appendix

## 7.1 Tools used

We used the following tools to produce this document:

- **LaTex** as typesetting system to write this document
- **LyX** as editor
- **Visio Professional** and **draw.io** to draw all the diagrams

## 7.2 Hours of work

| Date | Colaci | De Pasquale | Rinaldi |
|---|---|---|---|
| 16/01/17 | 2 | 2 | 1 |
| 17/01/17 | / | 1 | / |
| 18/01/17 | 3 | 3 | / |
| 19/01/17 | 2 | / | 2 |
| 20/01/17 | / | / | 2 |
| 21/01/17 | 2 | 3 | 2 |
| 22/01/17 | / | / | 2 |