POLITECNICO
MILANO 1863

Politecnico di Milano

A.A. 2016 – 2017

Software Engineering 2: "PowerEnJoy"

Project Plan

Marco Festa

January 22, 2017

Contents

1 Introduction

1.1 Purpose and Scope

The aim of this document is to analyze the whole system complexity and provide a scheduled plan to guide all of the implementation phase. Along with the implementation strategy costs, effort and possible risks are estimated for a better resource allocation. All of this data is unified to propose an adequate and punctual budget request. To better generate the project schedule two main approaches are adopted:

- *Function Points* approach to calculate project size.
- *COCOMOII* approach to estimate project cost and effort.

PowerEnJoy aim is to provide a car-sharing service for multiple cities with the peculiar characteristic of deploying exclusively electric cars. The scope of our project is to build a new digital management system to the company in order to support the service operations. A better description of the different goals to achieve is available in previous redacted documents (**RASD** and **DD**).

1.2 Definitions, Acronyms, Abbreviations

1.2.1 Definitions

Since not all of the system's aspects are mentioned in this document most of the definitions are omitted. For a better reference refer to previous documents (**RASD** or **DD**).

1.2.2 Acronyms

- **RASD**: Requirement Analysis and Specification Document
- **DD**: Design Document
- **ITPD**: Integration Test Plan Document
- **PP**: Project Plan Document
- **FP**: Function Points
- **ILF**: Internal Logic File
- **ELF**: External Logic File
- **EI**: External Input
- **EO**: External Output
- **EQ**: External Inquiries
- **CMM**: Capability Maturity Model
- **SLOC**: Source line of code
- **DBMS**: Database Management System
- **DB**: Database
- **UI**: user Interface
- **PGS**: Power Grid Station
- **API**: Application Programming Interface: a common way to communicate with other systems.
- **JEE**: Java Enterprise Edition
- **EUCARIS**: European Car and Driving License Information System

1.3 Reference Documents

- PowerEnJoy RASD
- PowerEnJoy DD
- PowerEnJoy ITPD
- Project description document: "Assignments AA 2016-2017.pdf"
- "COCOMO II – Model Definition Manual", version 2.1, 1995-2000, Center for Software Engineering, USC

2 Projecst size, cost and effort estimation

In this section are presented some detailed estimations of project size, cost and required effort. All of the main functionalities of PowerEnJoy system are considered and dissected with the Function Point approach to obtain a valid size estimation. For the cost and effort aspects we rely on the COCOMO approach.

2.1 Size estimation: function points

Function points are used to express and categorize all of the software functionalities. In the following table, we define how the various complexity levels of each function type are weighted in terms of function points.

	Complexity		
Function Type	*Low*	*Average*	*High*
Internal Logic Files	7	10	15
External Logic Files	5	7	10
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6

2.1.1 Internal Logic Files (ILFs)

All of the data structures and files managed by the application are considered Internal Logic Files. We provide a full list of all this data as it will be represented in the DB.

- **Login Data:** username, password.
- **User:** username, name, surname, date of birth, email, license ID, status, current ride, rides history, payment details, PIN.
- **Car:** id, status, plate, passengers, available, position, safe area, battery charge.

- **Ride:** ride id, status, car, reservation minutes, ride minutes, bill amount, bill status, start time, pickup position, drop off position.
- **PGS:** PGS id, position, max plugs, available plugs, cars connected.
- **Safe Area:** coordinates.
- **Payment Details:** payment option, credit card, bank account.

ILF	**Complexity**	**FPs**
Login Data	Low	7
User	Low	7
Car	Low	7
Ride	Low	7
PGS	Low	7
Safe Area	Low	7
Payment Details	Low	7
Total		49

2.1.2 External Logic Files (ELFs)

All data coming from external resources which needs to be handled by our system is considered an External Logic File.

- **Maps API Response:** data coming from the maps API on the user application (mobile or web) used to retrieve the current user position and display available cars on an interactive map.
- **Payment Api Response:** payment confirmation coming from the specific bank API.

- **Driving License Verification Response:** the connection to the EUCARIS system answers with a verification code confirming the driving license validity.

ELF	**Complexity**	**FPs**
Maps API Response	Low	5
Payment API Response	Low	5
Driving License Verification Response	Low	5
Total		15

2.1.3 External Inputs (EIs)

Listed below are all the functionality that need user input to interact with the system:

EI	**Complexity**	**FPs**
Registration	Average	3
Login/Logout	Low	2x3
Update User info	Average	4
Reserve Car	Average	4
Cancel Reservation	Average	4
Open Car	Average	4
Unlock Car	Low	3
End Ride	Low	3
Report Issue	Average	4
Total		35

2.1.4 External Inquiries (EQs)

External Inquiries are functionalities that provides information to the user after a query or request, retrieving data from an ILF or ELF.

EQ	**Complexity**	**FPs**
Search Cars with position/address	Average	4x2
Search PGS with position/address	Average	4x2
Get Rides History	Low	3
Get Car Info	Low	3
Total		22

2.1.5 External Outputs (EOs)

An external output is considered the action of the central system or one of its subcomponents to notify external agents (other components, administrator or the user itself):

EO	**Complexity**	**FPs**
Car Opened/Closed	Low	2x4
Car Locked/Unlocked	Low	2x4
Eco Mode notification	High	7
Total		23

2.1.6 Overall estimation

Function Type	**FPs**
Internal Logic Files	49
External Logic Files	15
External Inputs	35
External Outputs	23
External Inquiries	22
Total	***144***

The above table contains a summary of each different function point distribution. We can use the total amount of FPs to calculate the expected number of source lines of code (SLOC) using the formula:

$$SLOC = AVC \times FPs$$

Where AVC is a language dependent factor, in our case this can be approximated to a value of 45 (Java Enterprise Edition).

$$SLOC = 45 \times 144 = 6480$$

2.2 Cost and effort estimation: COCOMO II

COCOMO II approach is used below to estimate cost and effort of PoweEnJoy system.

2.2.1 Scale Factors

Five standard scale factors are considered in order to calculate the scale component E. If $E < 1.0$, the project exhibits economies of scale. If $E = 1.0$, the economies and diseconomies of scale are in balance. This linear model is often used for cost estimation of small projects. If $E > 1.0$, the project exhibits diseconomies of scale. This is generally because of two main factors: growth of interpersonal communications overhead and growth of large-system integration overhead.

Right below are shown the different weights assigned to each rating level of every different scale factor.

Scale Factor	**Rating Level**					
	Very Low	*Low*	*Nominal*	*High*	*Very High*	*Extra High*
PREC	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	5.07	4.05	3.04	2.03	1.01	0.00
RESL	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	7.80	6.24	4.68	3.12	1.56	0.00

Scale factors:

- **Precedentedness (PREC):** This factor expresses the past experience of our team with large scale projects.
  Our dev-team has a certain experience with software development but lacks of a business-oriented background. Our PREC level is set to *Nominal*.

- **Development flexibility (FLEX):** This factor expresses the allowed or possible flexibility during development process. This factor depends on how strictly goals and features were set by the company.
  Our flex factor is rated as *Nominal*.

- **Architecture / Risk Resolution (RESL):** This factor is rated High if the system architecture has been carefully developed considering possible risk.
  Since the systems operates with high value resources and sensitive data risk management covers all of the possible scenarios and thus has been rated as *High*.

- **Team Cohesion (TEAM):** This scale factor accounts for the possible sources of project turbulence and entropy because of difficulties in synchronizing the project's users, customers, developers or maintainers.
  Our team has already worked together and demonstrated high synergy and cooperation during software development. The factor rating is *High*.

- **Process Maturity (PMAT):** Reflects the process maturity of the organization.
  We can consider our process managed in accordance with agreed-upon and clear methods. Therefore, it is reasonable to assign a High rating to the scale factor.

The following table summarizes the chosen scale factor ratings.

Scale Factor	**Rate Level**	**Value**
PREC	Nominal	3.72
FLEX	Nominal	3.04
RESL	High	2.83
TEAM	High	2.19
PMAT	High	3.12
Total		***14.90***

2.2.2 Cost Drivers

Effort requirement is calculated through the use of post-architecture cost drivers since the system architecture has been clearly specified in the Design Document. All of the 17 different cost drivers range from a level of *Very Low* to *Very High*. Effort multipliers are extracted from the standard COCOMO II tables (Refer to COCOMO II Model Manual).

Product Factors:

- **Required Software Reliability (RELY):** 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 a slight inconvenience, then RELY is very low. If a failure would risk human life, then RELY is very high.
  Our system does not control any of the critical car hardware (breaks, accelerator, lights), but still has to guarantee no excessive distraction to the driver during his ride. Our value is set to *Nominal*.

- **Database Size (DATA):** This cost driver attempts to capture the effect large test data requirements have on product development.
  For PowerEnJoy system we had already estimated an amount of around 6600 SLOC. The test database size is instead estimated to be around 100MB which makes the D/SLOC ratio > 1000 meaning a value of *Very High*.

- **Product Complexity (CPLX):** Complexity is divided into five areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations.
  COCOMO standard tables suggest and average product complexity to the final ratio value of *Nominal*.

- **Developed for reusability (RUSE):** This cost driver accounts for the additional effort needed to construct components intended for reuse on current or future projects.
  We have no specific reusability constrain or future planned projects. We can opt for a value of *Nominal*.

- **Documentation Match to Life-Cycle Needs (DOCU):** This cost driver takes into account the required documentation level.
  Nominal is a balanced value for this specific cost factor.

Platform Factors:

- **Execution Time Constraint (TIME):** This factor takes into account the execution time constraint imposed upon a software system. It is calculated as the percentage of available execution time expected to be used by the system or subsystem consuming the execution time resource.

Following the COCOMO II guidelines, since our system requires a highly responsive and efficient software, we estimate a *High* rating.

- **Main Storage Constraint (STOR):** This rating represents the degree of main storage constraint imposed on a software system or subsystem.
  The moderate data-intensiveness nature of our project implies a reasonable value of *Nominal*.

- **Platform Volatility (PVOL):** In our scenario our platform is considered to be the Java Enterprise Edition stack which is updated in average once every year.
  The correct value to be set is *Low*.

Personnel Factors:

- **Analyst Capability (ACAP):** 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.
  Our team perfectly adjusts to a driver value of *Nominal*.

- **Programmer Capability (PCAP):** Evaluation is based on the capability of the programmers as a team rather than as individuals. Our dev-team ability allows us to confidently chose a *Very High* value for this driver.

- **Personnel Continuity (PCON):** The rating scale for PCON is in terms of the project's annual personnel turnover.
  This driver is set to *Very High*.

- **Applications Experience (APEX):** The rating for this cost driver is dependent on the level of applications experience of the project team developing the software system or subsystem.
  Our lack of experience imposes us to select a *Low* level.

- **Platform Experience (PLEX):** This driver considers the productivity influence of platform experience.
  Since our confidence with the supported platform is limited (databases, GUIs, networking), we account a *Nominal* value for the driver.

- **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. 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.
  Nominal is the correct value for this driver since our solid experience with Java programming language is balanced with the extremely basic knowledge of the JEE framework.

Project Factors:

- **Use of Software Tools (TOOL):** The tool rating ranges from simple edit and code, very low, to integrated life-cycle management tools, very high.
  IDE software like Eclipse and NetBeans, which we plan to use during development, certainly fall under the COCOMO *Very High* classification.

- **Multisite Development (SITE):** Determining this cost driver rating involves the assessment and judgement-based 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).
The entire team lives in the same city and can also be considered highly experienced with communication and file sharing tools. Factor is labeled *High*.

- **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. There are no specific constraints set on development period so the value is weighted *Nominal*.

Cost Drivers Summary:

The table summarizes all of the drivers' levels and presents the corresponding value in accordance with the COCOMO II manual.

Cost Driver	Level	Value
RELY	Nominal	1.00
DATA	Very High	1.28
CPLX	Nominal	1.00
RUSE	Nominal	1.00
DOCU	Nominal	1.00
TIME	High	1.11
STOR	Nominal	1.00
PVOL	Low	0.87
ACAP	Nominal	1.00
PCAP	Very High	0.76
PCON	Very High	0.81
APEX	Low	1.10
PLEX	Nominal	1.00
LTEX	Nominal	1.00
TOOL	Very High	0.78
SITE	High	0.93
SCED	Nominal	1.00
Result		**0.61**

2.2.3 Effort equation

The Post-Architecture COCOMO model formula estimates the necessary development effort.

Person-Month (PM) effort needed:

$$PM = A \times Size^{E} \times \Pi_{i=1}^{17} EM_i$$

$$E = B + 0.01 \times \sum_{j=1}^{5} SF_j$$

Where:

$A = 2.94$ is a fixed constant.

$B = 0.91$ is a fixed constant.

EM_i are the effort multipliers.

SF_j are the scale factors

Size is the estimated thousands of source lines of code (KSLOC)

$$E = 0.91 + 0.01 \times 14.90 = \mathbf{1.05}$$

$$PM = 2.94 \times 6.48^{1.05} \times 0.61 = \mathbf{12.74\ person\text{-}month}$$

The scale exponent is slightly greater than 1 implying diseconomies of scale.

2.2.4 Schedule estimation

Time to develop ($TDEV$) can be estimated as follows:

$$TDEV = C \times PM^{F}$$

$$F = D + 0.2 \times (E - B)$$

Where:

$B = 0.91$ is a fixed constant.

$C = 3.67$ is a fixed constant.

$D = 0.28$ is a fixed constant.

Results:

$$F = 0.28 + 0.2 \times (1.05 - 0.91) = 0.30$$

$$TDEV = 3.67 \times 12.74^{0.30} = 7.9 \text{ months}$$

3 Schedule

In this section is presented the high-level schedule for each fundamental phase of the project. Software development start date may be rescheduled, time slots and sub-tasks schedule is to be considered accurate.

- **RASD Writing:**

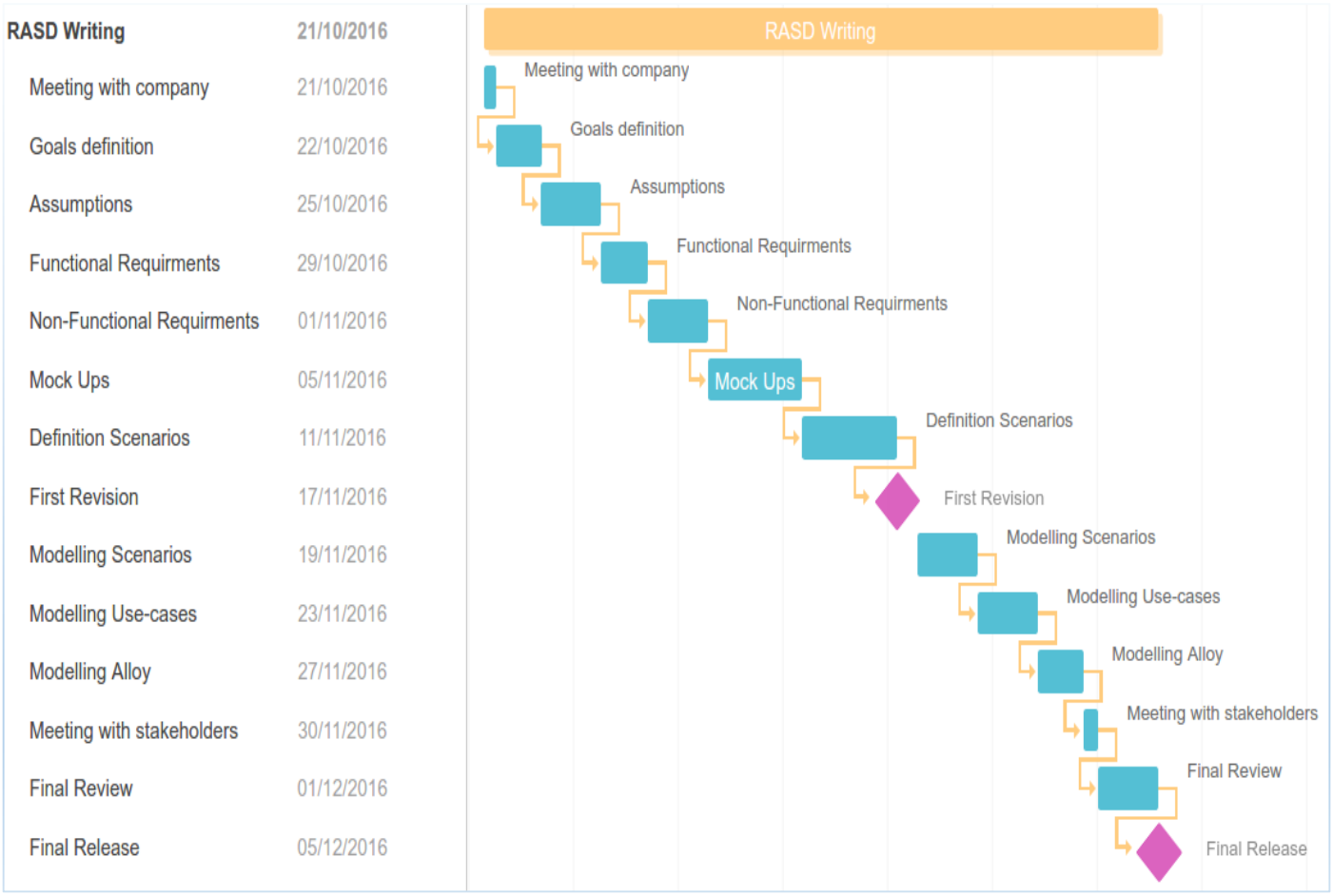

Figure 1: RASD Writing Gantt chart

- **DD Writing:**

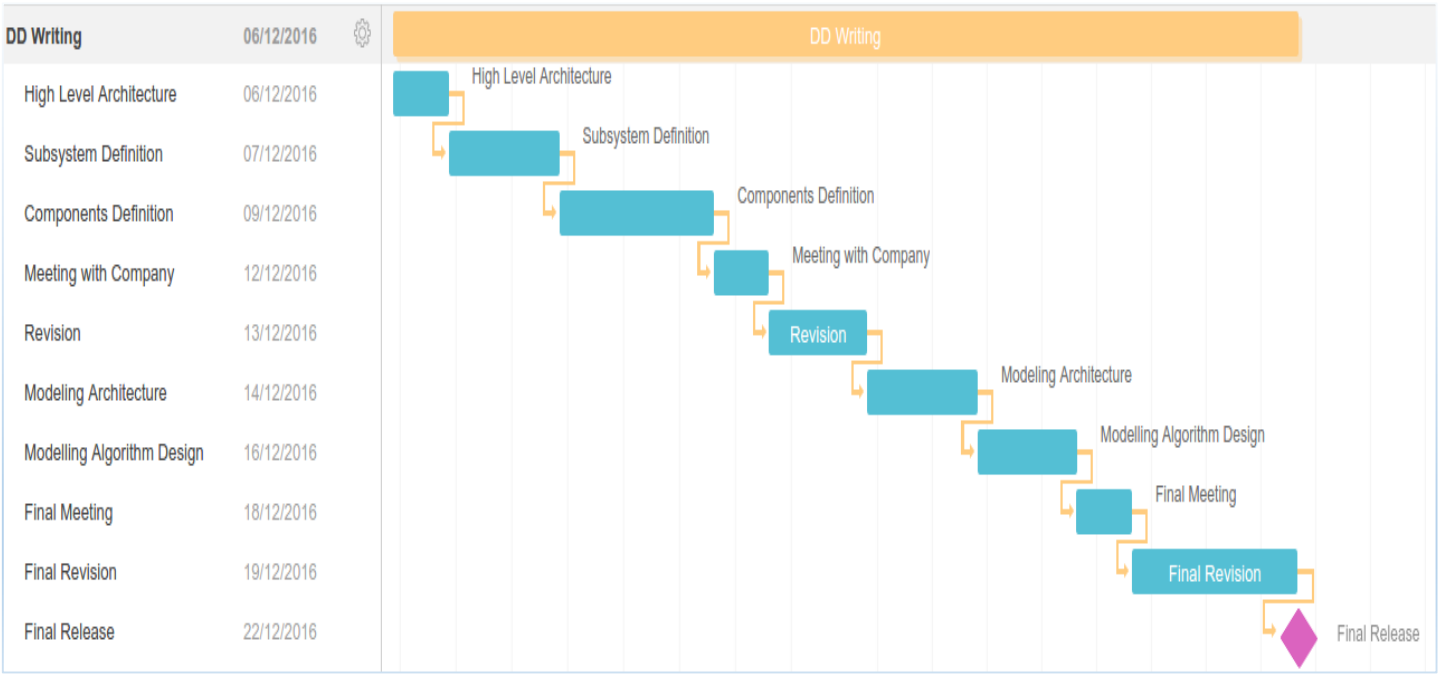

Figure 2: DD Writing Gantt chart

- **ITPD Writing:**

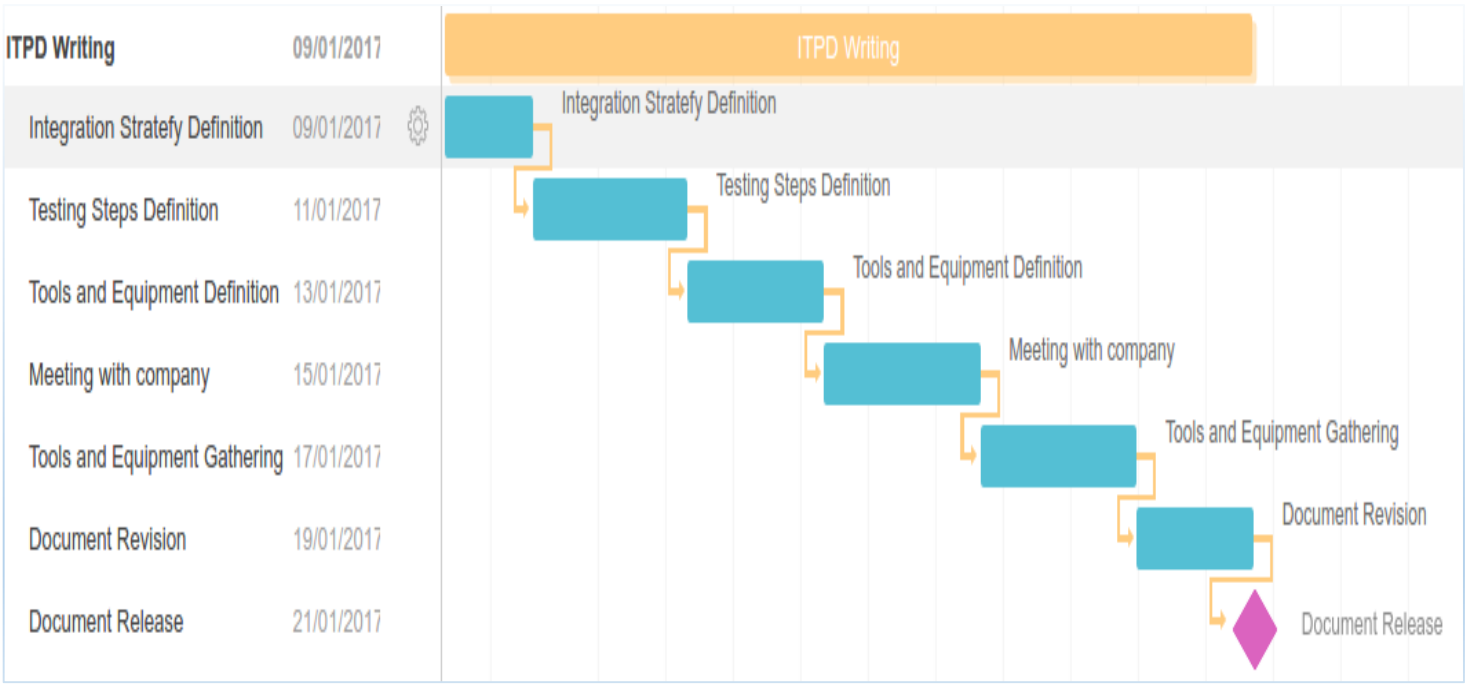

Figure 3: ITPD Writing Gantt chart

- **Development – Code Inspection – Testing:**

Figure 4: Development, Code Inspection, Testing Gantt chart

- **Deployment:**

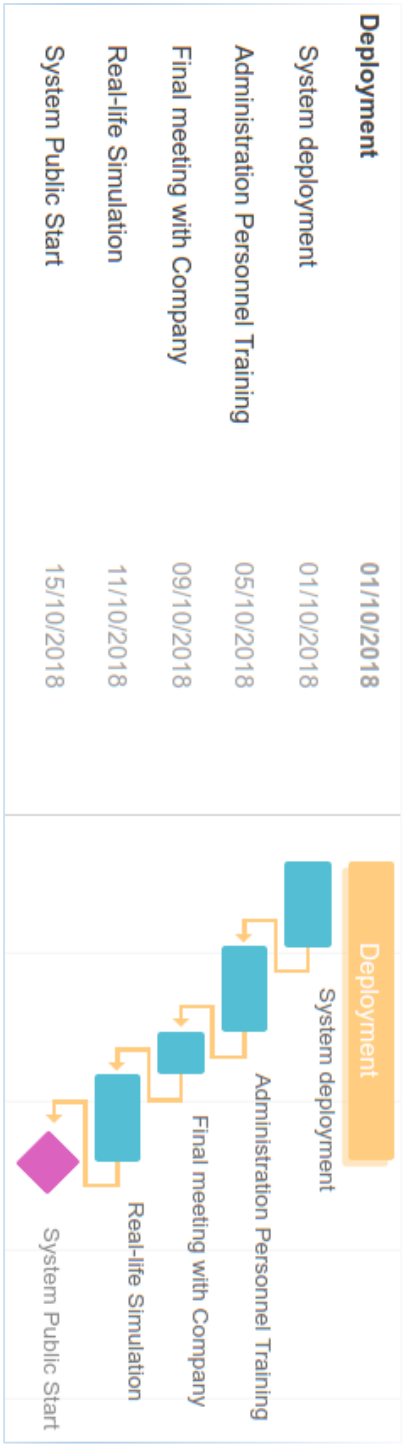

Figure 15: Deployment Gantt chart

4 Resource Allocation

All of the above presented tasks are assigned in this chapter to the various members of the team:

Marco Festa	
RASD Writing	All Document
DD Writing	All Document
ITPD Writing	All Document
Development and Testing	Application Subsystem development and testing, Car Subsystem Development and testing, Interfaces development, validation, code inspection, meetings with Company.
Deployment	All phases

Marco Sesta	
RASD Writing	None
DD Writing	None
ITPD Writing	None
Development and Testing	User applications (Mobile and Web) development and testing.
Deployment	All phases

Marco Testa	
RASD Writing	None
DD Writing	None
ITPD Writing	None
Development and Testing	Application Subsystem development and testing, Car Subsystem Development and testing, Interfaces development, validation, code inspection, meetings with Company.
Deployment	None

5 Risk Management

In this section are identified and addressed a numerous range of possible risks the whole project could face during all the phases of its development. Each risk is categorized for its nature and quantified for its probability and predicted impact. We later propose a possible strategy to minimize both of this two values.

5.1 Identified risks

Risk	Category	Probability	Impact
Wrong identified requirements	Project	Possible	Critical
Wrong predicted schedule	Project	Moderate	Critical
System complexity underestimate	Project	Unlikely	Critical
Personnel availability problems (illness, accidents etc.)	Project	Unlikely	Critical
Wrong or unstable hardware	Technical	Possible	Catastrophic
Low system performance	Technical	Possible	Critical
Lack of code documentation	Technical	Moderate	Low
Wrong estimated testing requirements	Technical	Unlikely	High
Wrong predicted budget	Business	Possible	High
Stakeholders commitment loss	Business	Possible	Critical
New car rental laws	Business	Unlikely	High

5.2 Risk strategies

In this section are proposed some possible prevention strategies and solutions to the listed risks. Most of this risks simply have their solution in a detailed and precise project analysis which technically has been already done during the redaction of the design and requirement documents (for this reason are omitted from the list below.)

Risk	Prevention	Solution
Wrong identified requirements	Perform monthly requirements validation	Identify missing requirements and fix RASD
Wrong predicted schedule	Monitor team performance and project development regularly	Modify immediately the schedule in order to avoid further delays
System complexity underestimate	Along with the internal project analysis perform external consultation	Modify schedule, consider team training
Personnel availability problems (illness, accidents etc.)	Nearly impossible to prevent	Hire more team members
Low system performance		Code optimization or higher hardware specs
Lack of code documentation	Impose rigid standard on code writing and perform monthly reviews	
Wrong predicted budget	Monthly meeting with stakeholders should prevent over budgeting	Propose budget raise

6 Appendix

6.1 Tools

- MS Word for the whole document.
- MS Excel for tables and formulas.
- Git for version control.
- GanttPRO (https://app.ganttpro.com/) for all the Gantt charts.

6.2 Effort spent

Marco Festa: 45 hours

6.3 Revisions

- PP v1.0 published January 22, 2017.