PowerEnJoy
Project Plan

Stefano Brandoli (mat. 875633)
Silvia Calcaterra (mat. 874887)
Samuele Conti (mat. 875708)

January 22, 2017

Version 1.0

Contents

1 Introduction

1.1 Revision History

Version	Date	Summary
1.0	22-01-2017	Initial Release

1.2 Purpose and Scope

This document is the Project Plan document for PowerEnjoy. The aim of this document is to provide information about the size, the cost and the effort estimation needed for the design and the implementation of the PowerEnjoy system. This document can also be useful as a guidance to define the required budget, the resource allocation and the schedule of the various activities.

1.3 Definitions, Acronym, Abbreviations

1.3.1 Definitions

- Person-Month = is the amount of time one person spends working on software development for one month. In COCOMO II this is normally assumed to be 152 person-hours.
- Maturity Model = set of structured levels that describe how well the behaviors, practices and processes of an organization can reliably and sustainably produce required outcomes.

1.3.2 Abbreviations

- UFP = Unadjusted Function Points
- ILF = Internal Logical File
- EIF = External Interface File
- EI = External Input
- EO = External Output
- EQ = External Inquiry
- PM = Person-Month
- SLOC = Source lines of code
- KSLOC = Kilo source lines of code.

1.4 List of Reference Documents

- Specification Document of PowerEnjoy.
- PowerEnjoy RASD.
- PowerEnjoy DD.
- Cocomo II: Model Definition Manual
  http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf
- Project Plan Example Document: myTaxiService
- UFP to SLOC conversion ratios:
  http://www.qsm.com/resources/function-point-languages-table
- Wikipedia: https://en.wikipedia.org/wiki/Capability_Maturity_Model

2 Size Estimation: Function Points

The Function Points approach provides an estimation of the size of a project considering the amount of functionalities to be developed and their complexity. This approach, according to the COCOMO II Manual, is based on the definition of the User Function Types (EI, EO, ILF, EIF, EQ) and follows this high level schema:

1. Count the functionalities of the system according to every User Function Type
2. Determine Complexity Levels
3. Calculate Complexity Weights
4. Calculate UFPs and relate them to SLOCs

In the next sections we will mainly follow the above mentioned schema.

2.1 Internal Logical Files

An internal logical file represents a set of data of the same type that is handled (generated and used) by the software system. Based on the Database Layer description in the PowerEnjoy DD we've identified:

- User
- RegisteredUser
- Operator
- Reservation
- Car Status
- SafeArea
- PowerSafeArea

2.2 External Interface Files

An external interface file represents data which is used by the system but generated by external applications or services. For our system we've identified:

- Maps Service: Google Maps
- BillingService: Stripe
- Vehicle Interface

2.3 External Inputs

An external input is an elementary operation that elaborates the data entering the external boundaries of the system. For our system we've identified:

- User registration
- User login
- Unlock Reserved Car
- Request Car reservation
- Terminate Car reservation
- Make a Stop during a Car reservation
- Enable Money Saving Option
- Request Car Maintenance
- Change Car Availability Status

2.4 External Outputs

An external output is an elementary operation that produces the data that leave the external boundaries of the system. For our system we've identified:

- Send commands to a PowerEnjoy car
- Send payment notification

2.5 External Inquiry

An external inquiry is an elementary operation regarding both input and output, without significant elaboration of data from logic files. For our system we've identified:

- Find available cars by GPS position or address
- Find unavailable cars

2.6 Complexity Levels

The following COCOMO II tables show how to classify each functionality of the system into the appropriate Complexity Level, based on the data element types contained and the number of file types referenced.

ILF/EIF	**Data Elements**		
Record Elements	**1-19**	**1-19**	**51+**
1	Low	Low	Avg
2-5	Low	Avg	High
6+	Avg	High	High

EO/EQ	**Data Elements**		
Record Elements	**1-5**	**6-19**	**20+**
0 or 1	Low	Low	Avg
2-3	Low	Avg	High
4+	Avg	High	High

EI	**Data Elements**		
Record Elements	**1-4**	**5-15**	**16+**
0 or 1	Low	Low	Avg
2-3	Low	Avg	High
3+	Avg	High	High

The following subsections will summarize using some tables the Complexity Level analysis that we've performed on our system, based on the COCOMO II tables for Complexity Levels just presented.

2.6.1 Internal Logical Files (ILFs)

Function	**Complexity**
User	Average
RegisteredUser	High
Operator	Average
Reservation	High
Car	High
Status	Low
SafeArea	High
PowerSafeArea	High

2.6.2 External Interface Files (EIFs)

Function	Complexity
Maps Service	High
BillingService	High
VehicleInterface	High

2.6.3 External Inputs (EIs)

Function	Complexity
User registration	Average
User login	Average
Unlock Reserved Car	Low
Request Car reservation	High
Terminate Car reservation	High
Make a Stop during a Car reservation	High
Enable Money Saving Option	Average
Request Car Maintenance	High
Change Car Availability Status	Average

2.6.4 External Inquiries (EQs)

Function	Complexity
Find available cars by GPS position or address	High
Find unavailable cars	High

2.6.5 External Outputs (EOs)

Function	Complexity
Send commands to a PowerEnjoy car	Average
Send payment notification	Low

2.7 Complexity Weights

The following COCOMO II table shows how to weight each User Function Type based on its Complexity Level. The following subsections will show the results of this analysis performed on our system.

	Complexity-Weight		
Function Type	**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

2.7.1 Internal Logical Files (ILFs)

Functions	Complexity	FP
User	Average	10
RegisteredUser	High	15
Operator	Average	10
Reservation	High	15
Car	High	15
Status	Low	7
Safe Area	High	15
Power Safe Area	High	15
Total		**102**

2.7.2 External Interface Files (EIFs)

Functions	Complexity	FP
Maps Service	High	10
BillingService	High	10
VehicleInterface	High	10
Total		**30**

2.7.3 External Inputs (EIs)

Functions	**Complexity**	**FP**
User registration	Average	4
User login	Average	4
Unlock Reserved Car	Low	3
Request Car reservation	High	6
Terminate Car reservation	High	6
Make a Stop during a Car reservation	High	6
Enable Money Saving Option	Average	4
Request Car Maintenance	High	6
Change Car Availability Status	Average	4
Total		**43**

2.7.4 External Inquiries (EQs)

Functions	**Complexity**	**FP**
Find available cars by GPS position or address	High	6
Find unavailable cars	High	6
Total		**12**

2.7.5 External Outputs (EOs)

Functions	**Complexity**	**FP**
Send commands to a PowerEnjoy car	Average	5
Send payment notification	Low	4
Total		**9**

2.8 Overall Estimation: Relating UFPs to SLOC

The following table summarizes the computation of the UFPs for our system.

User Function Type	**UFP**
Internal Logical Files	102
External Interface Files	30
External Inputs	43
External Inquiries	12
External Outputs	9
Total	**196**

The total count of UFPs calculated above is here converted in lines of code. Considering JEE 7 as the main development platform and according to the "UFP to SLOC conversion Ratios" (see Reference Documents), the average number of LOCs per UFP for JEE 7 is 46, so:

$$SLOC = 46 \cdot UFP = 9016$$

3 Cost and effort estimation: COCOMO II

In this section the COCOMO II approach will be used to estimate the cost and effort required to develop the PowerEnjoy system.

3.1 Scale Drivers

Each scale factor has a range of rating levels: from Very Low to Extra High. In order to evaluate the values of the scale drivers, we will then refer to the COCOMO II manual.

- **Precedentedness (PREC):** it reflects the amount of previous experience that the team has with the development of large scale projects. Since we are not field experts and most of the concepts used for this project are largely new to us, this value will be Low.
- **Development flexibility (FLEX):** it describes the amount of flexibility we have in the development process. The process has followed quite strictly a Waterfall lifecycle model and the document deadlines were fixed from the beginning. However, since we were free to use specific frameworks and devices to implement the system, this value will be Nominal.
- **Architecture / Risk Resolution (RESL):** it describes the level of awareness and details of the risk analysis performed. Since we will provide a general but also quite extensive risk analysis in a dedicated section of this document, this value will be set to High.
- **Team Cohesion (TEAM):** it takes into account the difficulties of synchronization of the stakeholders of the project, in particular it describes how well the team members can cooperate and interact with each other. Since our team has been highly cooperative, for us the value is Very High.
- **Process Maturity (PMAT):** it reflects the process maturity of the organization. Since we've followed a sets of defined and documented standard processes, and we think that the project goals have been successfully fulfilled, this value will be CMM Level 3 - Defined, which corresponds to High in the COCOMO II manual.

Scale Driver	**Factor**	**Value**
Precedentness (PREC)	Low	4.96
Development flexibility (FLEX)	Nominal	3.04
Architecture / Risk Resolution (RESL)	High	2.83
Team Cohesion (TEAM)	Very High	1.10
Process Maturity (PMAT)	High	3.12
$\sum SFi$		**15.05**

3.2 Cost Drivers

The Cost drivers are used to capture characteristics of the software development that affect the effort to complete the project. Since we're dealing with a Post-Architecture Analysis, these are the following cost drivers to consider:

Code	**Name**	**Factor**	**Value**
RELY	Required Software Reliability	High	1.10
DATA	Database size	Nominal	1
CPLX	Product Complexity	Nominal	1
RUSE	Required Reusability	Low	0.95
DOCU	Documentation match to life-cycle needs	Nominal	1
TIME	Execution Time Constraint	High	1.11
STOR	Main Storage Constraint	High	1.05
PVOL	Platform Volatility	Nominal	1
ACAP	Analyst Capability	Nominal	1
PCAP	Programmer Capability	Nominal	1
APEX	Application Experience	Very Low	1.22
PLEX	Platform Experience	Very Low	1.19
LTEX	Language and Tool Experience	Low	1.09
PCON	Personnel Continuity	High	0.90
TOOL	Usage of Software Tools	Nominal	1
SITE	Multisite Development	High	0.93
SCED	Required Development Schedule	High	1
Total	$EAF = \prod Ci$		**1.613**

3.3 Effort Equation

This equation is used to evaluate the effort required, measured in Person-Months:

$$Effort = A \cdot EAF \cdot (KSLOC)^E$$

Where:

- $A = 2.94$, approximates a productivity constant in PM/KSLOC for COCOMO II
- $EAF = 1.613$, product of all cost drivers, as calculated above
- $KSLOC = 9.016$, evaluated in the worst case
- $B = 0.91$ for COCOMO II
- $E = B + 0.01 \cdot \sum SF = 15.05 = 1.0605$

The total effort is evaluated as 48.84 person-months, so we can approximate it to **49 person-months**.

3.4 Schedule Estimation

The duration it will take to develop the system is estimated by the following formula:

$$Duration = C \cdot (PM)^F$$

Where:

- $PM = 49$, as calculated above
- $B = 0.91$, for COCOMO II
- $C = 3.67$, for COCOMO II
- $D = 0.28$, for COCOMO II
- $E = 1.0605$, as calculated above
- $F = D + 0.2 \cdot (E - B) = 0.3101$

The duration calculated is 12.27 months, which can be approximated in the worst case to **13 months**, which seems to be a reasonable estimate.

4 Tasks and Schedules

The project can be provided with a high-level schedule that defines adequate deadlines for each of its phases. By generally following the standard waterfall schema, it's possible to identify these main tasks:

- **Deliver the Requirement Analysis and Specification Document (RASD)**, in which are described the functional and non-functional requirements of the project, along with the goals, the domain assumptions, a list of possible scenarios, model and runtime views regarding the system under examination.
- **Deliver the Design Document (DD)**, describing the design and the architecture of the system.
- **Deliver the Integration Testing Plan Document (ITPD)**, containing the main strategies adopted to test the integration between the main components of the systems.
- **Deliver the Project Plan (PP)**, whose scope is described in the Introduction of this document.
- **Develop and implement the project**, preferably in parallel with the unit testing of the components.
- **Perform integration testing** of the main components of the system.
- **Deploy the system** in the real world.

Notice that independently from the level of depth adopted in the identification and description of the requirements, the customer of the project could request new features to be introduced along the different tasks.

In general, the executional order between the tasks must be respected, although some flexibility is encouraged to handle more easily the development of the system: it is advisable, in fact, to back up the develop phase with the unit testing of the components in order to not lose too much effort and resources in late corrections and modifications of the software.

As evaluate using the COCOMO II model in the previous section, the project will last 13 months, starting from October 2016 and finishing in November 2017.

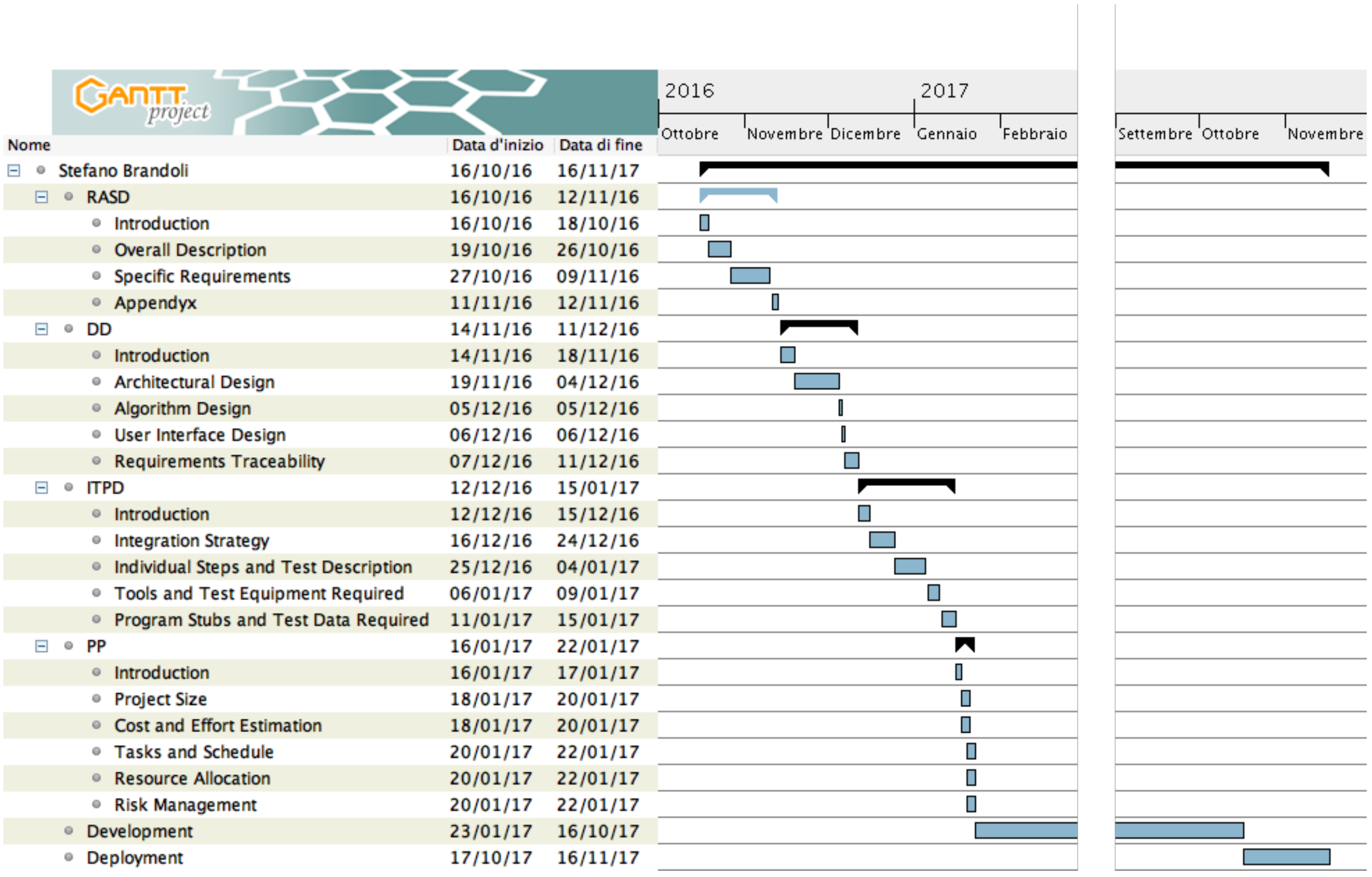
Task	**Starting Date**	**End Date**
RASD	16/10/2016	13/11/2016
DD	14/11/2016	11/12/2016
ITPD	12/12/2016	15/01/2017
PP	16/01/2017	22/01/2017
Development	23/01/2017	16/10/2017
Deployment	17/10/2017	16/11/2017

5 Resource Allocation

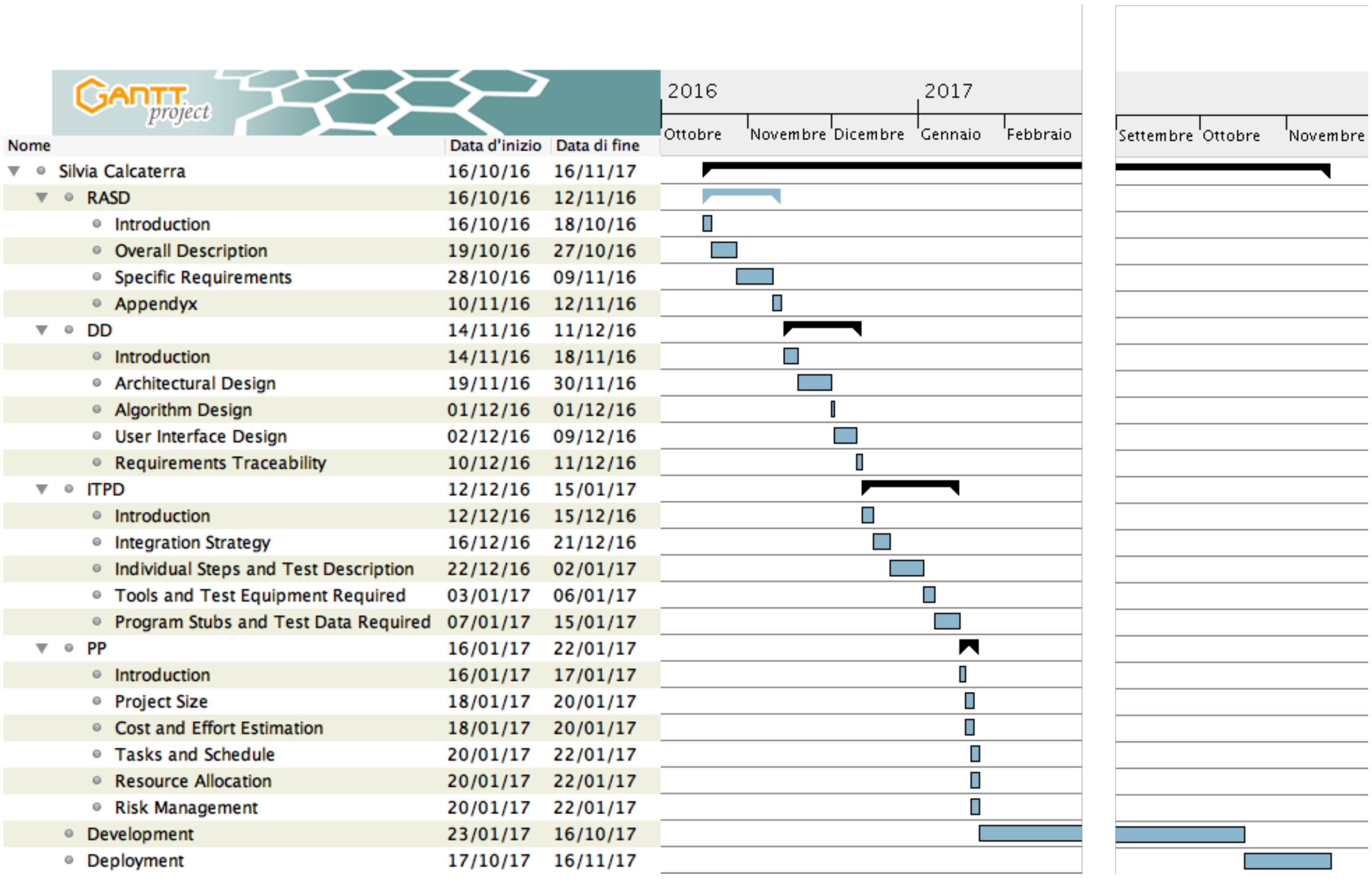
In this chapter the different tasks defined in the previous section are divided between the members of the team to provide a high-level overview of the personal workload required by this project. Notice that the resource allocation is not described at lower details, since the dimension of the team is fairly small compared to the one of the project: refined resource allocation will be introduced directly during each task of the project.

The main tasks of the project are divided in some main subtasks, corresponding to the main sections of the related documents. As explained more thoroughly in the risk section of the document, it's advisable that each member of the team gives his contribution to each project task, in order to avoid any possible inconvenience caused by a sudden unavailability of a member in a small team: a wider insight of the project could slow down the general process but would definitively provide more robustness.

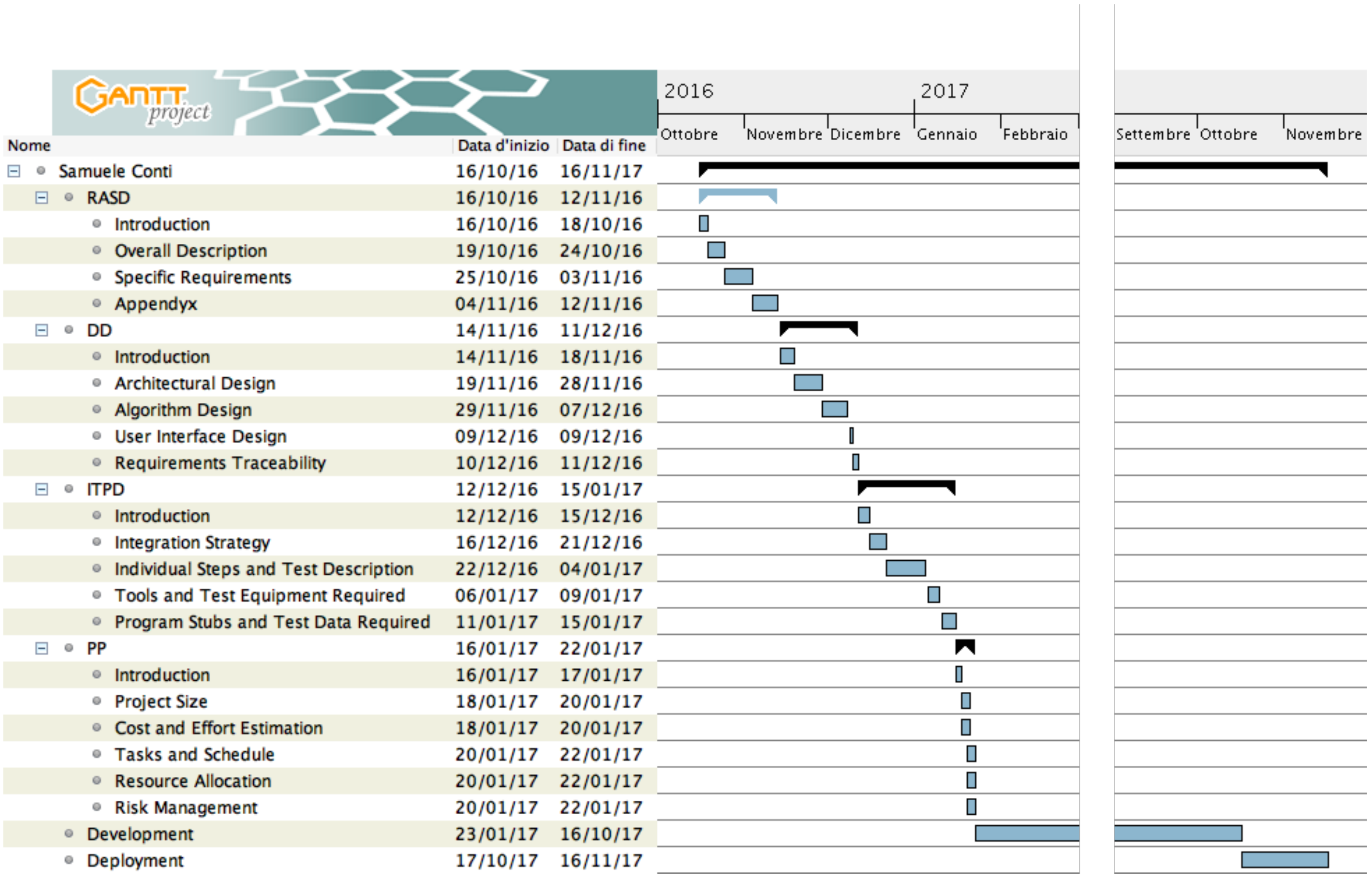
Notice that in the pictures below part of the Development stage was "cut out" to be able to fit the images in the page and to focus better on the first phases of the projects.



Nome	Data d'inizio	Data di fine
Stefano Brandoli	16/10/16	16/11/17
RASD	16/10/16	12/11/16
Introduction	16/10/16	18/10/16
Overall Description	19/10/16	26/10/16
Specific Requirements	27/10/16	09/11/16
Appendyx	11/11/16	12/11/16
DD	14/11/16	11/12/16
Introduction	14/11/16	18/11/16
Architectural Design	19/11/16	04/12/16
Algorithm Design	05/12/16	05/12/16
User Interface Design	06/12/16	06/12/16
Requirements Traceability	07/12/16	11/12/16
ITPD	12/12/16	15/01/17
Introduction	12/12/16	15/12/16
Integration Strategy	16/12/16	24/12/16
Individual Steps and Test Description	25/12/16	04/01/17
Tools and Test Equipment Required	06/01/17	09/01/17
Program Stubs and Test Data Required	11/01/17	15/01/17
PP	16/01/17	22/01/17
Introduction	16/01/17	17/01/17
Project Size	18/01/17	20/01/17
Cost and Effort Estimation	18/01/17	20/01/17
Tasks and Schedule	20/01/17	22/01/17
Resource Allocation	20/01/17	22/01/17
Risk Management	20/01/17	22/01/17
Development	23/01/17	16/10/17
Deployment	17/10/17	16/11/17



Nome	Data d'inizio	Data di fine
Silvia Calcaterra	16/10/16	16/11/17
RASD	16/10/16	12/11/16
Introduction	16/10/16	18/10/16
Overall Description	19/10/16	27/10/16
Specific Requirements	28/10/16	09/11/16
Appendyx	10/11/16	12/11/16
DD	14/11/16	11/12/16
Introduction	14/11/16	18/11/16
Architectural Design	19/11/16	30/11/16
Algorithm Design	01/12/16	01/12/16
User Interface Design	02/12/16	09/12/16
Requirements Traceability	10/12/16	11/12/16
ITPD	12/12/16	15/01/17
Introduction	12/12/16	15/12/16
Integration Strategy	16/12/16	21/12/16
Individual Steps and Test Description	22/12/16	02/01/17
Tools and Test Equipment Required	03/01/17	06/01/17
Program Stubs and Test Data Required	07/01/17	15/01/17
PP	16/01/17	22/01/17
Introduction	16/01/17	17/01/17
Project Size	18/01/17	20/01/17
Cost and Effort Estimation	18/01/17	20/01/17
Tasks and Schedule	20/01/17	22/01/17
Resource Allocation	20/01/17	22/01/17
Risk Management	20/01/17	22/01/17
Development	23/01/17	16/10/17
Deployment	17/10/17	16/11/17



Nome	Data d'inizio	Data di fine
Samuele Conti	16/10/16	16/11/17
RASD	16/10/16	12/11/16
Introduction	16/10/16	18/10/16
Overall Description	19/10/16	24/10/16
Specific Requirements	25/10/16	03/11/16
Appendyx	04/11/16	12/11/16
DD	14/11/16	11/12/16
Introduction	14/11/16	18/11/16
Architectural Design	19/11/16	28/11/16
Algorithm Design	29/11/16	07/12/16
User Interface Design	09/12/16	09/12/16
Requirements Traceability	10/12/16	11/12/16
ITPD	12/12/16	15/01/17
Introduction	12/12/16	15/12/16
Integration Strategy	16/12/16	21/12/16
Individual Steps and Test Description	22/12/16	04/01/17
Tools and Test Equipment Required	06/01/17	09/01/17
Program Stubs and Test Data Required	11/01/17	15/01/17
PP	16/01/17	22/01/17
Introduction	16/01/17	17/01/17
Project Size	18/01/17	20/01/17
Cost and Effort Estimation	18/01/17	20/01/17
Tasks and Schedule	20/01/17	22/01/17
Resource Allocation	20/01/17	22/01/17
Risk Management	20/01/17	22/01/17
Development	23/01/17	16/10/17
Deployment	17/10/17	16/11/17

6 Risk Management

The aim of this section is to identify the risks related with our system and to analyze their sphere of influence and probability to occur, in order to try to reason on strategies to tackle them (Proactive Approach). We distinguish between the following risk categories:

- Project Risks
- Technical Risks
- Business Risks

6.1 Project Risks

Project risks are related to the project plan. If they occur usually the whole project schedule shifts forward and the cost of the project increases.

- **Schedule Delays:** A wrong project schedule definition is likely to lead to delays in the delivery of the finished product. This risk can be prevented by being cautious about the schedule total duration: we've already overestimated the total expected duration of the project to take this into account.
- **Underestimated development time:** The development time for our system has been estimated in section 2, however a real implementation may require some unexpected extra time. In case this happens, we'll deliver a simplified working version of the system with less functionalities depending on the delay accumulated.
- **Requirements change:** Our customer might partially change the requirements during the development of the software. This risk cannot be prevented, but its effects can be mitigated by using traceability to help locating and distinguishing the various requirements, since understanding the difference between the old and the new requirements will be critical. In addition, writing reusable code will ease modifications if needed.
- **Problems among team members:** Problems inside the team, like synchronization and communication problems. The team will have to synchronize the work using tools for version control, however misunderstandings of interfaces and parameters used may lead to time loss and all the team members may not stay synchronized. This type of risk can be mitigated by establishing a clear division of responsibilities and tasks among the team members.
- **Staff illness:** Team members might get ill during the development of the project, causing potential delays in the delivery of the project. This problem can be prevented by ensuring that every team member knows what the other team members have been doing, so that the ill member can be temporarily replaced.

6.2 Technical Risks

Technical risks are related to the quality and timeliness of the project and if they occur the implementation of the project might become difficult or impossible.

- **Scalability issues:** our system might not be able to face an increase in the number of users using the PowerEnjoy services in a proper way. This problem can be prevented by using load balancers and machine replication.
- **Loss of data:** a loss of cars and users data in our DBMS might affect our system and its quality. This problem can be prevented by replicating the cars and users data, namely by keeping a backup copy of the most critical data of our system.
- **Integration failure:** our system might fail the integration tests described in the ITPD document; this risk can oblige us to rewrite pieces of code and to reorganize the interaction between components. This risk can be prevented by avoiding a "Big Bang Approach" for integration and by precisely defining in advance the interfaces of the components.
- **Issues with external services:** our system highly depends on some external services, namely Google Maps API, Stripe API and the Vehicle Interface. If some of these external services experiences problems that compromise its functionalities, our system's efficiency and quality might be jeopardized. We can prevent this risk by carefully choosing reliable and well known external services that are highly available.
- **Technical death:** the mobile operating systems that our system uses might be discontinued, obliging us to rewrite chunks of code, mainly regarding the front end, and this can highly increase the cost of maintenance for our system. This risk cannot be truly prevented, but it can be mitigated by choosing to use reliable, well known mobile operating systems.

6.3 Business Risks

Business risks are related to the viability of the system to be developed, if they occur they'll put it in danger in a general way.

- **Car provider bankrupt:** the provider of our cars may not be able to provide and maintain new vehicles because of a bankrupt. We can partially prevent this risk by choosing to rely on a well known car provided on the market and that is not likely to have serious economic issues.
- **Budget:** our customer may not be able to pay the whole project. This risk can be prevented by doing a correct and accurate feasible study before starting the real development of the project.
- **Legislation change:** the government of the city might decide to change the placement, size or rules of restricted traffic areas; this may obstacle

our system, since we will have to increment the fees of our users or write new code to forbid access to these areas.

- **System acceptance**
  - **No acceptance by city administration:** the city administration may not trust the innovative ideas of PowerEnjoy or decide that another competitor in the car sharing service is useless inside the city area. This may obstacole the development and success of the system itself.
  - **No acceptance by potential customers:** potential customers might decide not to trust the new PowerEnJoy service and they might continue to use the traditional transportation services inside the city. This risk can be prevented by creating an appealing and efficient service that uses innovative systems in order to attract the potential customers.
- **Competitors:** there are other famous car sharing services available in the city that can steal clients from our service. In order to deal with this risk, we might need to develop new and innovative features for PowerEnJoy and/or to make it more efficient with respect to the other car sharing services.

Risk	**Type**	**Probability**	**Effects**
Schedule delays	Project risk	Medium	Moderate
Underestimated development time	Project risk	Low	Moderate
Requirements change	Project risk	Low	Critical
Problems among team members	Project risk	Low	Critical
Staff illness	Project risk	Low	Moderate
Scalability issues	Technical risk	Medium	Moderate
Loss of data	Technical risk	Medium	Critical
Integration failure	Technical risk	Low	Critical
Issues with external services	Technical risk	Low	Critical
Technical death	Technical risk	Medium	Moderate
Car provider bankrupt	Business risk	Low	Critical
Budget	Business risk	Low	Critical
Legislation change	Business risk	Low	Moderate
System acceptance	Business risk	Low	Critical
Competitors	Business risk	High	Moderate

7 Software and tools used

- Git (https://github.com/) : for the version controlling of files shared between the team.
- Slack (https://slack.com/): used for group communication.
- GoogleDocs (https://www.google.it/intl/it/docs/about/): to write this document.
- GanttProject (http://www.ganttproject.biz/download): to create the schedule diagrams.
- Lyx (http://www.lyx.org/): to format this document.

8 Effort spent

Total hours of work for the PP creation:

- Stefano Brandoli: 9 hours
- Silvia Calcaterra: 6 hours
- Samuele Conti: 6 hours