Project Plan Document

POWER Enjoy

Authors:

Cannas, Castiglioni, Loiacono

POLITECNICO
MILANO 1863

SUMMARY

PART 1: INTRODUCTION

1.1 PURPOSE AND SCOPE

With the Project Plan Document (PPD), we aim at precisely describing the development planning for the accomplishment of our Power Enjoy software project.

It consists of four separate sections:

1. **Project Size Estimation**: in this section, we are going to use the Unadjusted Functional Points (UFP) and COCOMO II approaches in order to give an approximated estimation of the Power Enjoy project's size and effort cost;
2. **Task Identification**: in this section, we will explain how the data produced during the project size estimation phase has been used to identify all the activities necessary for the project fulfillment, starting from the Requirement Analysis and Specification (RASD) to the Integration Testing phase;
3. **Project Schedule**: a schedule with the assignment of the different tasks of the project to the various team's member will be carried out inside this section;
4. **Risk Identification and Management**: throughout this section, we will evaluate all the possible risks associated to the various project development phases, illustrating some possible recovery actions to be taken in order to correctly handle risk situations that could harm the project final achievement.

With all these information, we aim to offer a document of support for the decision of a Project Manager, and for the correct definition of resources needed and their allocation, and for budget required.

1.2 LIST OF ABBREVIATIONS

- **RASD:** Requirements Analysis and Specifications Document.
- **FP:** Functional Points.
- **ILF:** Internal Logic Files.
- **ELF:** External Logic Files.
- **EI:** External Inputs.
- **EO:** External Outputs.

- **EQ:** External Inquiries.
- **SLOC:** Source Lines Of Code.

1.3 REFERENCE DOCUMENTS

- Assignments AA 2016-2017.pdf
- Project Planning Example Document.pdf
- The COCOMO II Model Definition Manual (version 2.1, 1995 { 2000 Center for Software Engineering, USC}).
- Project Management Basics + Advanced Dec. 1.pdf
- PowerEnJoy_RASD.pdf
- “Function Point Counting Practices: Manual Release 4.0” International Function Point Users’Group (IFPUG), Blendonview Office Park, 5008-28 Pine Creek Drive, Westerville, OH 43081-4899
- http://www.qsm.com/resources/function-point-languages-table

PART 2: PROJECT SIZE AND EFFORT COST ESTIMATION

As we have briefly illustrated in the introduction part, the main focus of this section is on the illustration of the process of estimation of the project size (in terms of Source Line Of Codes) and of the effort cost for its achievement.

Determining the size of a project is crucial in its model estimation, even though this is not a trivial task. In the context of this document, we used the Unadjusted Function Point approach, as described in the COCOMO II Model Definition Manual (the source is the IFPUG 1994's release manual), and as explained during the course lectures, since the COCOMO II Model uses this metric in its effort cost evaluation.

We will first provide the analysis of the main functionalities of our system, divided by the main components implementing it (mobileApplication, Central Server, Car Application), following the requirements and specification illustrated in our RASD document; then, we will perform our effort cost evaluation, using the KSLOC computed during the first phase.

2.1 FUNCTIONAL POINT SIZE ESTIMATION

The UFP approach assesses a project size by quantifying the information processing functionality associated with 5 different types of functions.

These types of functions are:

- functions associated with major homogenous sets of internal managed data (**ILF, Internal Logic Files**);
- functions associated with major homogenous sets of external managed data (**ELF, External Interface Files**);
- functions associated with external control input data (**External Input**);
- functions associated with data generated for the external environment (**External Output**);
- functions associated with both input control data and output data for the external environment (**External Inquiries**).

For all of our three components, we divided their main functionalities (starting from the requirements specified in the RASD document) into these different function types, estimating the complexity associated with their development using proper statistical standardized tools, as defined by the IFPUG

Tables with the correct complexity estimation are provided below for each function type.

For Internal Logical Files and External Interface Files

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

For External Output and External Inquiry

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

For External Input

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

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

Figure 1: Function Types and Complexity-Weight tables

2.1.1 INTERNAL LOGIC FILES (ILFs)

For the Central Server Component functionalities, several different data types are required in order to provide an efficient service to the customer.

In first place, the Central Server has to store data about the Users connecting with and using the PowerEnJoy software: this means collecting information about their personal details (name, surname, E-mail, password, Id), driver license (license driver number), payment information (credit card number) and position. All of these information can be easily stored in a single relational table, leading to the definition of a logic file of low complexity.

The same reasoning lies behind the definition of the Car logic file: the system should be aware of the Cars' Id, their status (Available, NotAvailable and so on), their battery level (and their recharging status likewise), ride information (whether the engine is on or off, the number of passengers on the Car, the lock or unlock status of the doors) and finally of their position. As for the Users, these data can be stored in a single table, so we considered this file as a low complexity one.

The Reservation logic file instead is a little more complex one. Obviously, it has to store both the information about the User reserving the Car, and the Car being reserved: it is then a two level data structure, with a few more data about the time of the reservation, and whether the User picked up the Car or not. We set the complexity for this ILF to average.

For what concerns the SafeAreas, the Central Server needs of course to be aware of their position, the number of parkings (divided in total and free ones), their IDs, and whether they have a power grid for the Car's recharge. Again, this is a simple data structure, so we set this complexity to low.

Lastly, surely the Central Server needs to keep track of the amount of money charged to the User; thus, we thought that a logic file with the history of all the User's payments (User information along with the type of charging and the amount of it) was needed, though that is a low complexity file too.

ILFs	**Complexity**	**FPs**
User data	Low	7
Car data	Low	7
Reservation data	Average	10
SafeArea data	Low	7

Payment data	Low	7

Table 1: Central Server Internal Logic Files (ILFs)

Moving on to the Car Application, the Internal Logic File needed for its functioning is merely a data structure containing the Car's ID and its status.

In fact, while we would have the application still managing the other data about position, battery level, etc…, this operation is going to happen dynamically, having the data in fact stored and provided to the Application by the Car's Electric Control Unit (ECU), through the GMContainter of the GreenE-Box framework.

Therefore, we have a single ILF of low complexity containing the Car's static data.

ILFS	Complexity	FPs
Car's static data	Low	7

Table 2: Car Application Internal Logic Files (ILFs)

Finally, regarding to the MobileApplication, this component will surely have to store information about the User, in a similar fashion to what the Central Server does, and about the Reservations the User made. However, we do not the MobileApplication to store all the information about the Car being reserved, leading to the a ILF of lower complexity inside of it, we just information about the time of Reservation and the reserved Car's actual position.

We will thus have two ILFs of low complexity in both case.

ILFs	Complexity	FPs
User	Low	7
Reservation	Low	7

Table 3: MobileApplication Internal Logic Files (ILFs)

2.1.2 EXTERNAL INTERFACE FILES

In this subsection we are going to analyze the external data sources used by the three main high-level components of the PowerEnJoy system in order to perform their functionalities.

In this analysis, we have considered as external data source both the data requested through APIs and interfaces to external service providers, and the data exchanged between our high-level components during their normal deployment.

In this interactions (both between the Central Server and the Applications, and between the components and the external services), we suppose to use the RESTFul API with data exchanged in JSON and XML format.

The Central Server exploits primarily EIFs from external service providers during the Reservation and Registration processes, when verifying the User's credit car plafond and the validity of its driver license. Without going in further details about the analysis of this type of services, we suppose that the actual responses provided by the Stripe and Eucaris API are simple JSON/XML response, so we set their complexity both to Low.

We have more information about the Car's dynamic info, which is used by the Central Server in order to control the Car's usage during a ride and for updating the Car's data inside the DBs. These data comprehend battery level, position of the Car, and low level information about its functioning (whether the engine is on, doors are lock, etc…). Since these are simple data types, we set to Low the overall complexity of this ELF.

EIFs	**Complexity**	**FPs**
Credit card plafond	Low	5
Driver license info	Low	5
Car dynamic data	Low	5

Table 4: Central Server External Interface Files (EIFs)

Switching to the CarApplication, the main EIFs it uses are the Car's dynamic data (as provided by the Car's ECU in order to monitor the Car's global status), and the SafeAreas' data during the report of a low battery emergency (provided by the Central Server), when the Application requests the nearest available SafeAreas in order to indicate the User possible spots where recharging the Car. Both of them are simple data structure with simple fileds, so we set their complexity to low.

EIFs	Complexity	FPs
Safe Area positions	Low	5
Car's dynamic data	Low	5

Table 5: CarApplication External Interface Files (EIFs)

Finally, the MobileApplication since it has been conceived as a thin client, relies heavily on external interface files provided by both the Central Server and external service providers.

In first place, it uses the SafeArea, Payment and Car data structure, as designed for the Central Server: all of the have low complexity, and they still have from the MobileApplication point of view.

The only average complexity file, is the one made of the GoogleMaps from the Google Maps API, used for the displacement of the SafeArea search results on the mobileApplication screen.

EIFs	Complexity	FPs
Safe Area	Low	5
Payment	Low	5
Car	Low	5
GoogleMap	Average	7

Table 6: MobileApplication External Interface Files (EIFs)

2.1.3 EXTERNAL INPUTS

The PowerEnJoy software should perform several functionalities using external inputs as main data. For what concerns the Central Server, it surely has to provide functions allowing the Users to login and reserve a Car for their rides. The first is not a really complex operation, while the second involves the updating of several data in different data structure, thus, it can be considered as an average complexity operation.

External inputs	**complexity**	**FPs**
Login	Low	3
Reservation	Average	4

Table 7: Central Server External Inputs

The CarApplication instead performs a single operation on an external input, that is the charging computing for the User's rental of the Car, based on data provided by the Car's ECU on the status of the engine (on or off) and of the Car's position (inside or outside a Safe Area). This is still not a complex operation, so its complexity is set to low.

External Input	**Complexity**	**FPs**
ComputeCharging	Low	3

Table 8: Car Application External Inputs

Lasty, the MobileApplication provides two external inputs actions, that are the notification of payments and reservations' expirations on the Central Server inputs. Both of them are simple data operations of low complexity.

External Inputs	**Complexity**	**FPs**
DisplayExpirationNotification	Low	3

DisplayPaymentNotification	Low	3

Table 9: MobileApplication External Inputs

2.1.4 EXTERNAL INQUIRIES

Since the environment where the PowerEnJoy software is going to be deployed requires a good percentage of interaction between our different components, the external inquiries we detected are several and characterize all of our high-level software applications.

The Central Server performs various operations involving both input and output. The first of them, is the total payment charging, that needs the data about the Car's status and the payment information of the User, involving as output operation the actual charging operation and the payment notification to the User. This is actually a not really complex operation, since involves just the User ID and payment method information, and the Car's dynamic data.

The search operation of available Cars for reservation can be considered as an external inquiry too, but instead its execution is not trivial, since it has to select a list of SafeArea with available Cars for reservation inside a certain range of position: we set its complexity as average.

The MoneySaving search operation can be considered as a similar operation to the search, so we set the same complexity of it.

The low battery emergency handle is an operation similar, which involves a search of SafeAreas for the recharging of a Car starting from its position: for this reason we set its complexity to average too.

Finally, the Registration operation has been considered as an inquiry too, since it involves the sent of an E-mail as a feedback for the User with its password. However, in this case we considered the operation as not complex.

External Inquiries	**Complexity**	**FPs**
PaymentCharging	Low	3
Search operation	Average	4
MoneySaving search operation	Average	4

LowBattery emergency handle	Average	4
Registration	Low	3

Table 10: Central Server External Inquiries

The CarApplication involves several external inquiries too: the report of a low battery emergency (as seen for the Central Server), and the report of a general emergency too. From the CarApplication's point of view still, this are not really complex operation, involving the manipulation of few input data types, so we set their complexity as low.

The lock and unlock operation has been considered a external inquire in this case too, since it involves the exchange of input and output data with the Server (and the MobileApplication in turn) as a feedback. Its complexity is low.

External Inquiries	**Complexity**	**FPs**
Lock/Unlock	Low	3
ReportLowBattery	Low	3
ReportEmergency	Low	3

Table 11: CarApplication External Inquiries

In short, the MobileApplication performs a certain number of external inquiries too, the same we saw from the Central Server side. These comprehend the search and MoneySaving search operations, the login, the reservation and registration, and the lock unlock of the Car.

However, they do not involve a great data elaboration, as they can be considered just as requests of services from the Server with the sending of few parameters, expecting some data in return. Their complexities are all set to low.

External Inquiries	Complexity	FPs
Search	Low	3
MoneySavingSearch	Low	3
Login	Low	3
Reservation	Low	3
Registration	Low	3
Lock/Unlock	Low	3

Table 12: Mobile Application External Inquiries

2.1.5 EXTERNAL OUTPUTS

The operations involving the generation of data for the external environment (external outputs) are limited in this case just to the Central Server component and the Car Application. The Mobile Application, thought a thin client, when generating data from the environment is always collocated inside the client-server paradigm, thus, expecting data in return and thus this operations can be thought as external inquiries (as we explained in the previous subsection).

For what concerns the Central Server, the external outputs operations are those involved in the mail sending for the requests of operators, and the Users' notification of expired reservations. These are all simple operations involving few data, their complexity has been considered as low.

External outputs	Complexity	FPs
mailSending	Low	4
NotifyExpiration	Low	4

Table 13: Central Server External Outputs

The CarApplication presents even a simplier set of external output functions, since the only output operation is that of the Users' directed information through a display, regarding the actual charging for the Car, and the display of the nearest SafeAreas in case of low battery emergency. This operation is of low complexity.

External Output	Complexity	FPs
DisplayInfo	Low	4

Table 14: CarApplication External Output

2.1.6 OVERALL ESTIMATION

In this section we will provide an overall estimation of the FPs so far calculated, and we will calculate the Effective Source Lines of Code (SLOC) for our software project.

For each component we will perform a separated computation, since the actual framework on which they are going to be developed requires a different programming language.

In fact, the Central Server will be developed using the JEE framework, while the Car and MobileApplication are based on the Android OS, thus requiring to be developed using the Java language.

For the Central Server we computed the overall FPs estimate:

Function Type	**Total FPs**
Internal Logic Files	38
External Interface Files	15
External Input	7
External Inquiries	18
External Outputs	8

The overall Server total SLOC are then = 8+18+7+15+38*JEEAvgGearingFactor(=46) = 3956 SLOC with an high scale factor (High = 67), which leads to an upper bound of 5762 SLOC.

For the CarApplication we computed the overall FPs estimate of:

Function Type	**Total FPs**
Internal Logic Files	7
External Interface Files	10

External Input	3
External Inquiries	9
External Outputs	4

The overall CarApplication total SLOC are then = 7+10+9+3+4 = 33*JavaAvgGearingFactor(=53)=1749 SLOC, with an upperbound using the JavaHighGearingFactor(=134) equal to 4422 SLOC.

Finally, for the MobileApplication we computed the overall FPs estimate is of:

Function Type	**Total FPs**
Internal Logic Files	14
External Interface Files	22
External Input	6
External Inquiries	9

The overall mobileApplication total SLOCs are then equal to = (18+6+22+14)*JavaAverageGearingFactor(=53) = 3180 SLOC, with an upperbound of 60*JavaHighGearingFactor(=134) = 8040 SLOC.

The overall average system SLOC are then equal to 8855 SLOC = 8.855 KSLOC, with an overall upperbound system SLOC equal to 18224 SLOC = 18.22 KSLOC.

Please note that all the gearing factors have been taken from the Function Point Language Table Version 5.0, from the Quality Software Management association.

2.2.1 SCALE DRIVERS

In this section we are going to provide an overview on the computation of the scale and cost drivers used for the evaluation of the effort costs in the COCOMO II model.

For each Scale Driver category:

- **Precedentedness**: there already exist some similar application. The PREC is set to High.
- **FLEX**: the FLEX is set to high because there aren't particular flexibility issue.
- **RESL**: the risk analysis is performed in moderate way. The RESL is set to low.
- **TEAM**: All the member of the team cooperate, The team cohesion is high.
- **PMat**: our PMat level is Level 1 (upper half). PMAT value is low.

Scale Drivers	**Factor**	**Value**
PREC	High	2.48
FLEX	High	2.03
RESL	Low	5.65
TEAM	High	2.19
PMAT	low	6.24
SUM		**18.59**

2.2.2 COST DRIVERS

In the evaluation of the Cost Drivers for the computation of the COMOMO II effort costs, we put ourselves in the case of an Early Design project, since there is no older project and or platform on which develop our work.

Thus, following the COMOMO II Manual, we evaluated all the Post Architecture cost drivers and then converted them into their early design counterparts. The following list and table explain all the reasoning behind our single choices for every driver's value.

- **Required Software Re-usability (RELY):** the system failure could produce a lot of problems to the clients that use the service, but it's reasonable to think the customers can easily find another way to move around the city. On the other hand the failure would surely be an economic loss for the company. For this reason the RELY cost driver is set to nominal.

- **Database size (DATA):** the expected database data size is about 100MB, since the project SLOC are 8'855 the ratio DB bytes/SLOCK is more than 11000. The DATA is set to very high.

- **Product Complexity(CPLX):** set to nominal according to the COCOMO II rating scale.

- **Required Usability(RUSE):** the reusability is only across project, so the RUSE cost driver is set to nominal.

- **Documentation match to life-cycle needs (DOCU):** every part of the life-cycle is covered by documentation, so the DOCU cost driver is set to Nominal.

- **Execution time constraints (TIME)**: the expected use of the CPU is related to the number of users that access to the service. We can expect an use on the available execution time greater than 70% of the overall time. So TIME driver is set to high.

- **Storage Constraints (STOR)**: a common Hard Disk can contain much more of the memory requested by the application, so the STOR is set to nominal.

- **Platform volatility (PVOL):** we don't expect frequent major change of the platform for this reason the PVOL is set to low.

- **Analyst Capability (ACAP)**: the Analysts who are partecipating to the project have Analysis and Design ability, efficiency and thoroughness, so the ACAP is set to high.

- **Programmer Capability (PCAP):** the programmers that participate to the project can communicate and cooperate and have medium programming ability, for this reason the PCAP is set to Nominal.

- **Application Experience (APEX):** the team does not have any past experience with application similar to project one,. For this reason the APEX driver is set to very low.

- **Platform Experience (PLEX):** the team does not have any past experience with JEE platform and car applications' programming, but little experience with databases and Android. The PLEX is set to low.

- **Language and Tool Experience (LTEX):** for the same reasons listed above for the PLEX cost driver, the LTEX is set to low too.

- **Personnel Continuity (PCON):** because the development time of the project is short(should be less than an year) the PCON is set to very high.

- **Use of Software Tools(TOOL):** the project is supported with mature lifecycle management tools, so the TOOL driver is set to High.

- **Multisite Development (SITE):** the project members live in the same area and can meet regularly. Moreover, the speed of Internet connection allows them to communicate remotely .So the SITE driver is set to very high.

- **Required Development Schedule (SCED):** we have no data regarding nominal schedules of projects requiring the same effort. Despite of this, the effort have been well distributed during the project lifetime, without any needs to accelerate the schedule. The SCED is set to Nominal.

	Post-Architecture cost drivers	value	sum	Early-design value
PERS	ACAP	4	**12**	**0.75**
	PCAP	3		
	PCON	5		
RCPX	RELY	3	**14**	**1.33**
	DATA	5		
	CPLX	3		
	DOCU	3		
RUSE				**1.00**
PDIF	TIME	4	**9**	**1.00**
	STOR	3		
	PVOL	2		
PREX	APEX	1	**5**	**1.33**
	PLEX	2		
	LTEX	2		
FCIL	TOOL	4	**9**	**0.73**
	SITE	5		
SCED				**1.00**

PRODUCT				**0.968**

2.2.3 EFFORT EQUATION

The estimated value in PM(person month) is measured with the equation:

$$\textbf{Effort} = \textbf{A} + \textbf{EAF} + \textbf{KSLOC}\textbf{\^{}}\textbf{E}$$

where:

- E= B +0.01* **Σ** SF
- A=2.94
- B=0.91
- EAF=product of cost drivers
- **Σ** SF=sum of scale factors

For our project, the estimated effort cost is equal to:

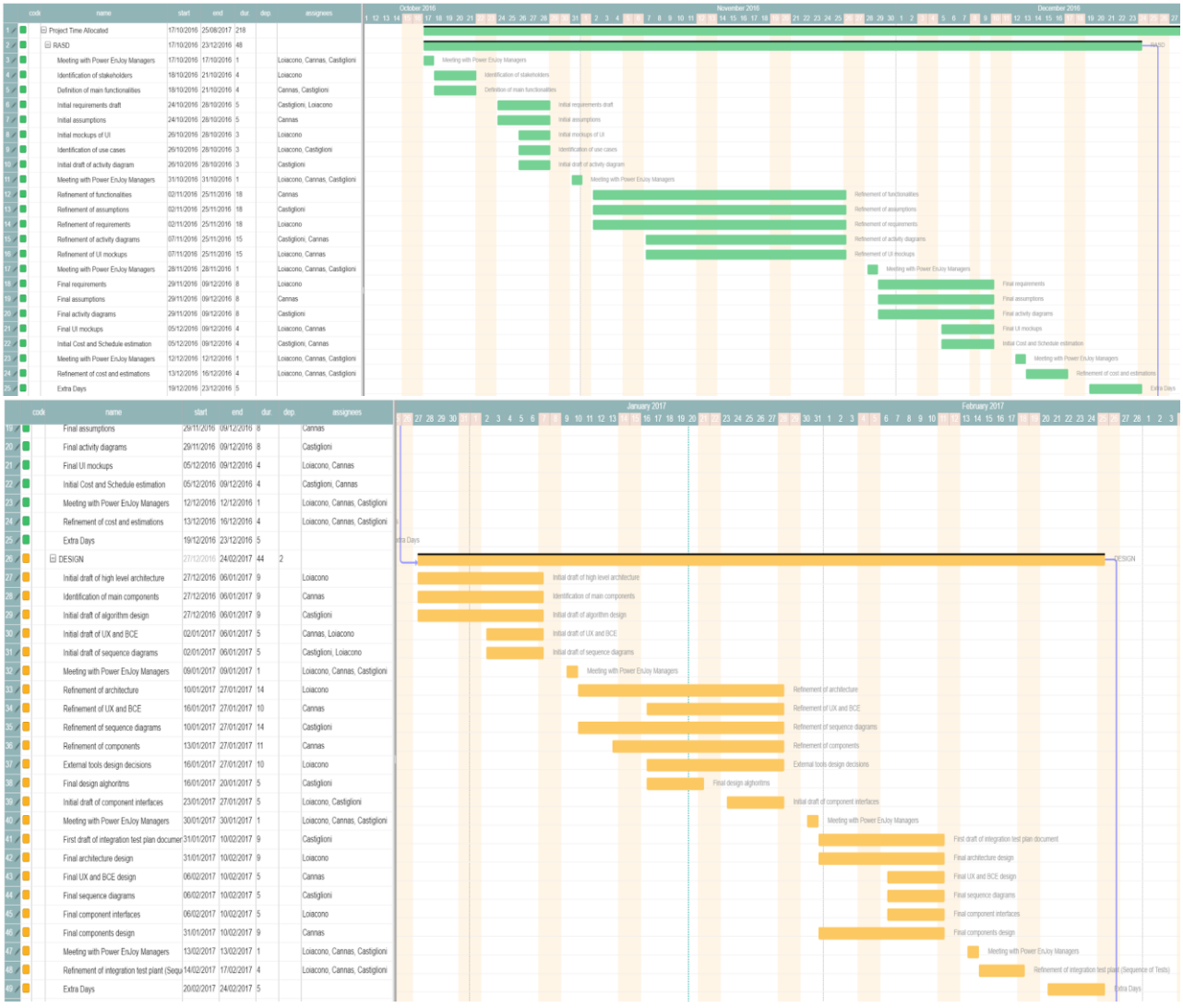
$$\textbf{Effort} = 2.94 * 0.968 * 8.855^{(0.91+0.01* 18.59)} = 31\text{PM}$$

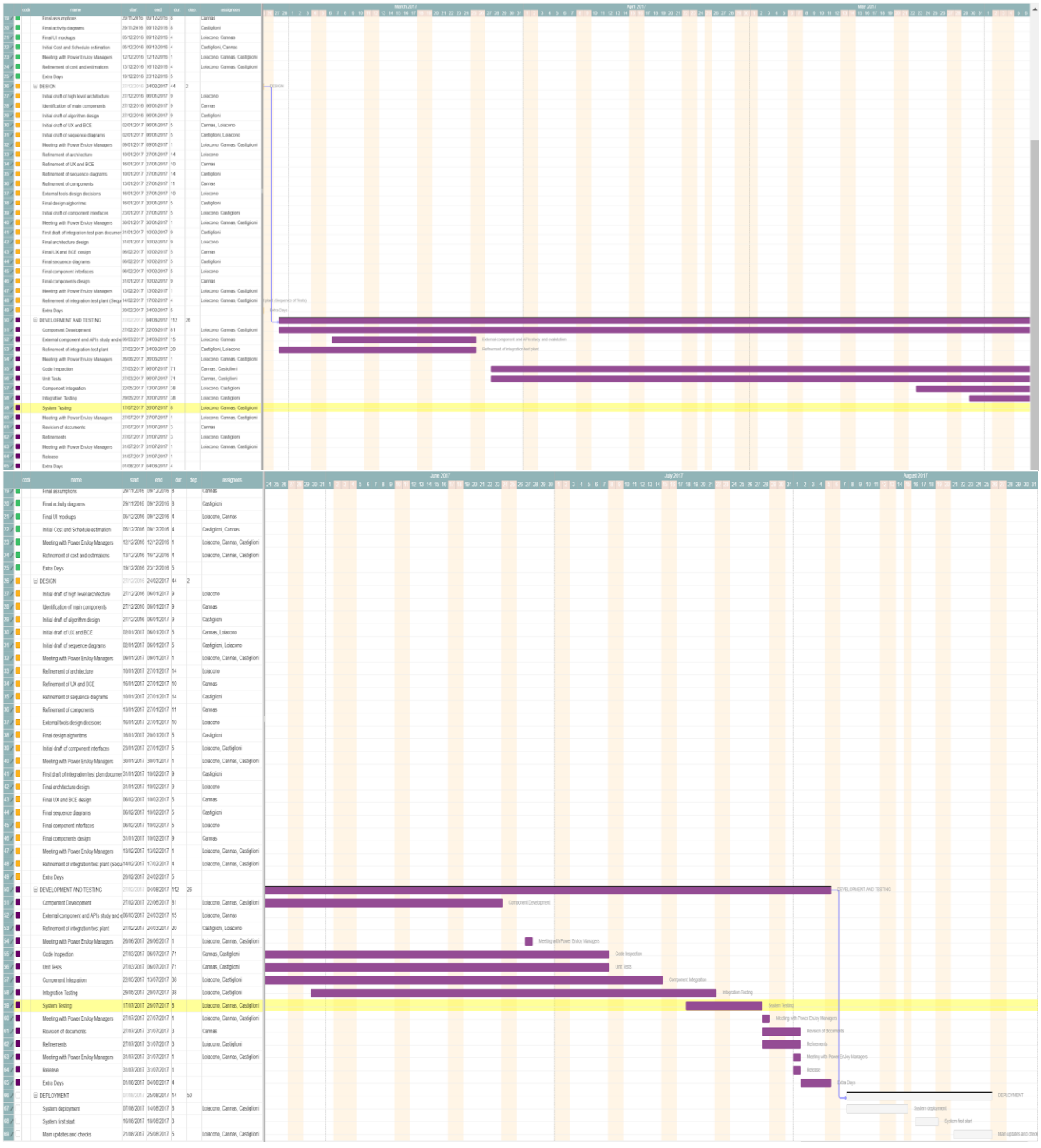
PART 3: SCHEDULE

In this chapter we are going to provide a general time schedule for our project, based on the results obtained using COCOMO II.

A Gantt chart, with all high level activities, is provided. Furthermore, to design a consistent planning, considerations made during the risk analysis have been taken into account: extra days and distributed responsibilities are part of the plan.

Although this is a project made only for education purposes, we tried to depict a real-world plan, adding tasks (such as meeting with Managers), which didn't happen in reality. The plan is also aware of holidays and weekends, and also tries to split tasks in the fairest way possible between components. Because of the length of the chart, it is divided in 4 different images, splitted in the same number of pages. Different colours highlight different subsections of the plan.





In order to give a better perspective, the single images of our Gannt charts are available in the project repository.

PART 4: RISK MANAGEMENT

In this section, we analyzed the main risks the project might encounter during its development. We are going to rank them relying on the Chance of Happening, and Impact on the Overall project. Every risk will be also categorized as:

- **Project Risk**: threatening the project plan and the schedule;
- **Technical Risk**: threatening the overall quality and implementation of the project;
- **Business Risk:** threatening the overall functions implemented, and the viability of the product.

Probability	Damage
Low Moderate High	Low Moderate High Serious Cathastrophic

For every risk with Probability equals to High and Impact greater or equal than High, and for every risk with Damage equal to Catastrophic, we also devised a possible strategy to be actuated fir avoiding massive damages to the project.

Risk Name	Unexpected lack of personnel
Category	Project Risk, Technical Risk
Probability	High
Impact	High to Catastrophic
Description	Because only three people are involved in the project, if one of them suddenly becomes unavailable, because of a disease, family business or worse (i.e. he call himself out of the project), the impact of this

	risk to happen is very high on the project schedule, which will be delayed, and on the technical development, because of the knowledge borrowed by the missing person.
Strategy	The main strategy to assess the technical risk related to a lack of personnel could be splitting responsibilities of people on various parts of the project, as well as always having a backup person who knows how to manage parts of the project related to someone else. To assess the project risk related to a missing person, the only viable solution seems to have some extra days inside the Gantt, in order to ensure that some possible delays are already taken into account. Another possible solution is to hire more people before the project even starts,

Risk Name	Low Budget
Category	Business Risk
Probability	Low
Impact	Serious
Description	The probability of this risk to happen is low, because of the use of COCOMO II and FPs in order to evaluate the time needed to develop the project. The impact instead is high, because a low budget will probably lead to major team defections, and lack of personnel: people will leave the project (and the company) if they are not paid. The estimation made in this document should prevent any of this scenarios to happen.

Risk Name	Wrong Functionalities implemented
Category	Project Risk, Business Risk
Probability	Low
Impact	Catastrophic

Description	If the RASD is not well produced, or the stakeholder's needs are not well documented and analyzed, it could be possible that a wrong product is going to be developed. While the probability of this scenario is low, because every member of the team has a Bachelor Degree in Computer Science and enough knowledge to tackle the problems posed by requirement engineering, the impact of this event would be catastrophic.
Strategy	A continuous communication with Stakeholders and a refinement of Requirements and Goals should prevent this risk to happen. Power EnJoy should always be involved in the development of the application.

Risk Name	Bad\Wrong Interfaces implemented
Category	Business Risk
Probability	High
Impact	Medium to High
Description	Because none of us has ever designed a User interface for commercial purpose, the probability of developing one that is of difficult comprehension and usage, is high. This would also have a medium to high impact on the satisfaction of the Stakeholders (primarily Power EnJoy), which could reject the project as unfinished, or not meeting the requirements.
Strategy	The main strategy to assess this risk is to have continuous test of the user interface by external people, providing them with questionnaires to fulfil in order to analyze the average User's experience with the use of our application. Using Material Design to implement UIs should avoid us from contacting an artist\designer. Also a constant communication with Power EnJoy's Managers should help us developing a User Interface meeting expectations.

Risk Name	Gold plating
Category	Project Risk
Probability	Low
Impact	Medium
Description	This risk is unlikely to happen, because of the time constraints imposed by the budget and by the schedule defined in a previous section of this document. However, in the case it would becomes real (i.e. during Requirement definition, or Design phase), it could assess our schedule of the project.

Risk Name	New/Changed Requirements
Category	Project Risk, Technical Risk, Business Risk
Probability	Medium
Impact	Medium to Catastrophic
Description	Because the car sharing's market is recent and continuously evolving, our main stakeholder (Power EnJoy), could easily refine, change or add new requirements during the first phases of the project. This event could have a different impact on the project, depending on which requirement/functionalities are requested.
Strategy	The main strategy to assess this risk is having a continuous communication (meetings, public tests, etc. etc.) with Power EnJoy managers, in order to inform them about the state of the project, clarify the number of functionalities requested (i.e. what we signed for), and have them to understand the constraints imposed by the development of a software project.

Risk Name	Bad\Changed APIs\External Components
Category	Project Risk, Technical Risk, Business Risk
Probability	High
Impact	High
Description	Because of our massive usage of external APIs and components (Maps APIs, Green E-Box, ecc. ecc), it's highly possible that a modification or a change of the terms and condition will lead to a delay of our project schedule, as well as affecting our Design and the budget allocated to external resources. Also if these external components are highly related to ours, this would lead to large chunks of code to be rewritten.
Strategy	In order to protect ourselves from this risk, we design our software keeping in mind possible APIs alternatives to the ones used. We should also try to follow good design principles which guarantee portability and independency of the various components.

Risk Name	Wrong assumptions on software\hardware quality
Category	Project Risk, Technical Risk, Business Risk
Probability	Medium
Impact	Serious to Catastrophic
Description	If some of the piece of hardware used, or the software developed unexpectedly does not meet the non-functional requirements addressed in the first phase of the project, this might lead to major delays, because new solutions will need to be found, in order to address the requirements. Also, changing implementations could lead to a loss of quality on the technical side (i.e. a Database too slow), or a change in the budget allocated for hardware\software resources.
Strategy	In order to protect ourselves from this, we should design our

	software in order to prevent any technical debt; if we find out that a piece of hardware does not meet our requirements, we should always have another commercial and well-known alternative ready to be exchanged with the defective one.

Risk Name	Legal Risks
Category	Business Risks
Probability	Medium
Impact	Low to Serious
Description	Because the car sharing's market is relatively young, this could lead to new law procedures, which might affect the requirements for our project. The impact of this risk is not easy to define, and very difficult to prevent: the only way to assess this eventuality is to constantly stay informed about the legal field our project relates to.

PART 5: HOURS OF WORK

Date	Edoardo Cannas	Matteo Castiglioni	Tommaso Loiacono
13/01/17	2h	2h	2h
16/01/17	2h	2h	2h
17/01/17			3h
18/01/17	2h	2h	
19/01/17	2h	2h	3h
20/01/17			
21/01/17	2h	2h	
Overall	10h	10h	10h

The overall effort for the fulfillment of this document has been of 30 hours on the whole.