Politecnico di Milano

Software Engineering II project:
PowerEnjoy

Project Plan Document

Gregori Giacomo and Ruaro Nicola

January 22, 2017
Version 1.0

Contents

Abstract

This document provides a detailed description of the Project Plan for the PowerEnJoy system. It is based on the RASD, DD and ITPD documents presented in the previous deliveries and must properly estimate the project complexity and provide risk management strategies, it should also be consultable as a schedule plan.

Introduction

1.1 Purpose

The purpose of this document is to provide a reasonable estimation for the development effort needed. In the first chapters the Function Point Analysis and COCOMO II methods are used for the project complexity and effort estimation. In the following chapters a possible schedule and a risk's analysis will be presented, providing a proper project planning and task assignment other than effective procedures for risk management.

1.2 Scope

PowerEnJoy is a car-sharing service based on mobile and web applications which should allow users to reserve vehicles and use them. The application must be developed in order to maximize the effectiveness of the task planning and risk management.

Function Points Estimation

In this chapter the Function Point Estimation model will be used to give a reliable estimation for the project complexity based on its functionalities.

2.1 Weighting function points

In the following table the weights used to correlate the Function Types to the development effort are presented.

Function Types	Weight		
Internal Logic 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

2.2 Internal Logic Files (ILFs)

Internal Logic Files are defined as the "homogeneous set of data used and managed by the application". For the PowerEnJoy project, based on the RASD analysis, we identified 9 logical groups of user data or control information:

User : The User table stores all the credentials and payment data for the users, we expect a relatively large amount of users for our system and considering that the number of fields is also really high we can assume its complexity is HIGH.

Vehicle : The number of vehicles in our system will be limited. The Vehicle table, though, contains a large amount of fields. Therefore we can consider its complexity AVERAGE.

Location : The Location table stores all the vehicles, safe areas and charging stations positions, since its fields are really limited and also its entries, we can consider its complexity LOW.

Safe Area : The Safe Area table stores the safe areas as points composing a polygon. The amount of safe areas is limited and the number of points needed to represent a safe area is relatively low, we can consider its complexity LOW.

Charging Station : The Charging Station table has a small amount of fields and probably the most limited amount of entries among the data tables composing our system. Its complexity is LOW.

Reservation : Our system will have to deal with a large amount of reservations, which have a fairly complex structure, for this reason its complexity is considered HIGH.

Ride : Similarly to reservations, the PowerEnJoy system will handle a large amount of ride's records, which have a complex structure. We therefore consider its complexity HIGH.

Behaviours : The Behaviours table has a limited number of entries and a simple structure as it is composed of a small number of fields. Therefore we can consider its complexity LOW.

Payment : The Payment table has a fairly complex structure and its entries are proportional to the Reservation and Ride entries. Its complexity is then considered HIGH.

ILF	Complexity	FPs
User	HIGH	15
Vehicle	AVERAGE	10
Location	LOW	7
Safe Area	LOW	7
Charging Station	LOW	7
Reservation	HIGH	15
Ride	HIGH	15
Behaviours	LOW	7
Payment	HIGH	15
Total		98

2.3 External Interface Files (EIFs)

External Interface Files are defined as the "homogeneous set of data used by the application but generated and maintained by other applications". We identified 3 groups for EIFs:

Notification Services : The files generated by the Notification Services are records for the outgoing notifications requested by the PowerEnJoy system. The amount of notification data is consistent but its structure is simple and does not require any computation (other than checking the successful delivery), their complexity is then considered LOW.

Payment Services : The files generated by the Payment Services are records for the outgoing payment requests and for the ingoing payments. The amount of notification data is consistent (proportional to the reservations) but its structure is simple and does not require any computation (other than checking the successful delivery/receipt), their complexity is then considered LOW.

Map Services : The files generated by the Map Services are typically for reverse geocoding requests. The amount of map data is average but considering that this type of data requires computation to become usable we can assume its complexity as HIGH.

EIF	Complexity	FPs
Notification	LOW	5
Payment	LOW	5
Map	HIGH	10
Total		20

2.4 External Inputs (EIs)

External Inputs Files are defined as the "elementary operations to elaborate data coming from the external environment".
The EIF are classified based on three different operations' categories:

2.4.1 Guest

Registration : This operation is not complex but needs a number of checks over the request's fields for validation purposes. Its complexity is AVERAGE.

Login : The login operation involves only the AuthenticationManager component (and just the User ILF), its complexity is therefore LOW.

2.4.2 User

Logout : The logout operation is similar to the login, it involves the same components and ILF and it's then considered of LOW complexity.

Manage Account : This operation is similar to the registration operation as it is an update request for the user's data. The complexity is AVERAGE since it needs to check and validate the request's fields.

Reserve Car: The reservation procedure is not complex but involves User, Reservation and Vehicle ILFs in addition to a number of check and notification procedures for expiration. Its complexity is therefore AVERAGE.

Release Reservation: Similarly to the reservation procedure (as it involves the same ILFs but not the same check and notification procedures), the release procedure has LOW complexity.

Start Ride: The start ride operation involves the RideController, ReservationManager and CarManager components and it relies on a number of different checks. Its complexity is AVERAGE.

End Ride: The car lock procedure is fairly complex and requires interaction with different components and a number of ILFs for validation checks (location must be inside safe-area), status update and payment requests. Its complexity is considered HIGH.

2.4.3 On-Board

Update Sensor Data: This operation includes the update of the vehicle's location, odometer, battery level, status, number of seats and number of passengers. Since it deals only with the CarManager component and the Vehicle ILF its complexity is considered LOW.

EI	Complexity	FPs
Registration	AVERAGE	4
Login	LOW	3
Logout	LOW	3
Manage Account	AVERAGE	4
Reserve Car	AVERAGE	4
Release Reservation	LOW	3
Start Ride	AVERAGE	4
End Ride	HIGH	6
Update Sensor Data	LOW	3
Total		34

2.5 External Outputs (EOs)

External Outputs are defined as the "elementary operations that generates data for the external environment".

Notify the creation of a new user : This procedure includes the generation of new credentials. Since it does not involve neither complex computation nor numerous data sources its complexity is LOW.

Notify the expiration of a reservation : This procedure includes the application of the expiration fee. Since it must deal with different ILFs (Vehicle, Reservation, Payment) and EIFs (Payment Services, Notification Services) its complexity is HIGH.

Notify final charges : This procedure includes the computation of alternative charge situations. It must read and write data from various internal and external data sources and deal with different external interface files (Notification Services and Payment Services). Its complexity is HIGH.

Notify nearby cars on request : This procedure includes the retrieval of cars based on their current location. It is based on an ordering algorithm executed on a really limited set of locations and since it does not have to deal with numerous internal or external data source its complexity is AVERAGE.

Notify errors : This includes the notifications for errors caused by various data-input procedures. Since its based on simple but numerous validation checks we can consider its complexity AVERAGE.

EO	Complexity	FPs
Notify the creation of a new user	LOW	4
Notify the expiration of a reservation	HIGH	7
Notify final charges	HIGH	7
Notify nearby cars on request	AVERAGE	5
Notify errors	AVERAGE	5
Total		28

2.6 External Inquiries (EQs)

External Inquiries are defined as the "elementary operations that involve input and output". For the PowerEnJoy project, based on the RASD analysis, we identified 9 logical groups of user data or control information:

Retrieval of profile informations : Since this procedure reads data from a single internal data source (User) and does not require any additional computation, its complexity is LOW.

Retrieval of map data : The map data is based upon numerous data sources (Vehicle, Safe Area, Charging Station), for this reason its complexity is considered AVERAGE.

Retrieval of current reservation data : Since this procedure retrieves data only from the Reservation ILF and does not carry out any further processing, its complexity is LOW.

Retrieval of ride history : Similarly to the profile informations retrieval and reservation data retrieval, this procedure interacts only with the Ride table without any further processing. Its complexity is therefore LOW.

EQ	Complexity	FPs
Retrieval of profile informations	LOW	3
Retrieval of map data	AVERAGE	4
Retrieval of current reservation data	LOW	3
Retrieval of ride history	LOW	3
Total		13

2.7 Overall Estimation

Based on the table presented in section 2.1 and on the reasoning described in the precedent sections, we can estimate the Unadjusted Function Points for the PowerEnJoy system:

Type	Value
ILF	98
EIF	20
EI	34
EO	28
EQ	13
Total	193

We can proceed estimating the total number of lines of code: JEE will be the programming language and for this language the upper and lower-bound conversion factors are, respectively, 46 and 67. For our estimation, the mobile, web and on-board computer applications are not considered since they can be thought as pure presentation with no business logic.

The computed lower-bound is:

$$SLOC = 193 * 46 = 8.878$$

The computed lower-bound is:

$$SLOC = 193 * 67 = 12.931$$

Effort Estimation

3.1 COCOMO Approach

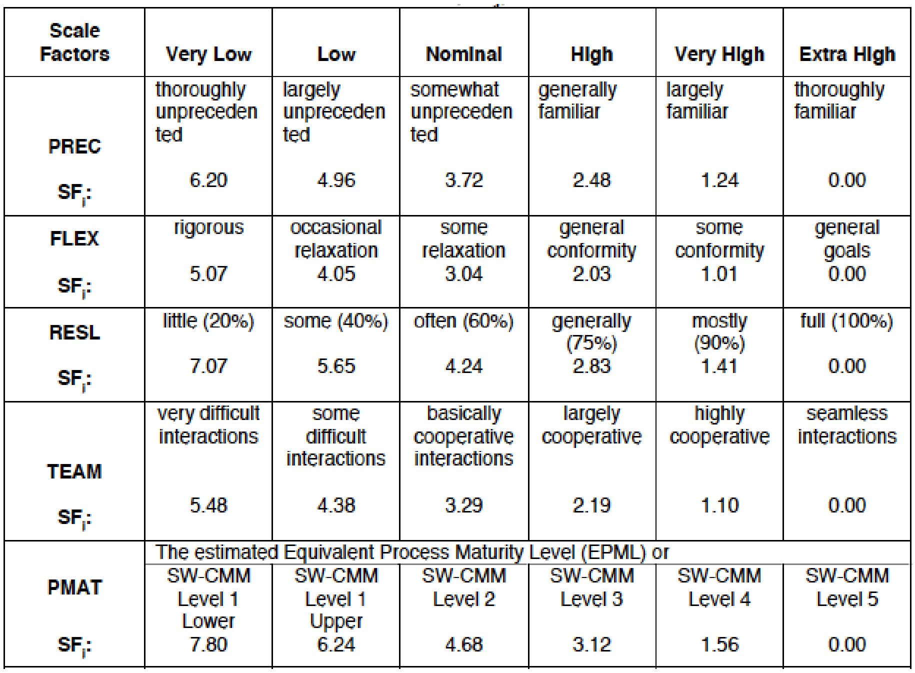
In this chapter, we use the COCOMO II model in order to estimate the effort needed to develop the PowerEnJoy system.
Scale Drivers and Cost Drivers are analysed in the following sections and are needed to calculate the Effort Equation, the main COCOMO formula:

$$\text{Effort} = A * EAF * KSLOC^{E} \text{ , where A = 2.94 .}$$

From the Scale Drivers we will obtain the value of E, while from the Cost Drivers we will obtain EAF.

3.2 Scale Drivers

In this section the Scale Factors table is used to obtain the values associated with the five main factors that characterize our Scale Drivers.



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

Precedentedness : This is the first project of this kind that our team developed, therefore the value will be set to LOW.

Development Flexibility : There are some instructions provided by the stakeholders but the specifics for the project are decided by us, therefore the value will be set to HIGH.

Architecture / Risk Resolution : We have a good risk management plan, the architecture is well-defined and the schedule is planned in an efficient way, therefore the value will be set to HIGH.

Team Cohesion : We were able to work as a team, sharing the same vision and working together, especially for the assignment's planning the document's checking, therefore this value will be set to VERY HIGH.

Process Maturity : The maturity of our project is assessed by the well know CMMI method and can be evaluated as Quantitatively Managed. The value will be set to NOMINAL.

Scale Factor	**Factor**	**Value**
Precedentedness	low	4.96
Development Flexibility	high	2.03
Architecture / Risk Resolution	high	1.41
Team Cohesion	very high	1.10
Process Maturity	nominal	4.68

TOTAL	14.18

3.2.1 Evaluation of the Exponent E

The equation that we use for the exponent's evaluation is:

E $= B + 0.01 * \sum_{j=1}^{5} SF_j$, where B = 0.91 .

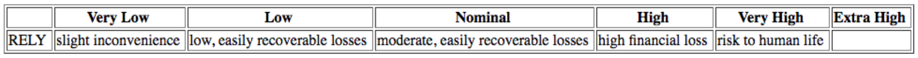
For our project $E = 0.91 + 0.01x14.18 \cong$**1.052**

3.3 Cost Drivers

These are the 17 effort multipliers used in COCOMO® II Post-Architecture model to adjust the nominal effort and reflect the software product under development. The ratings related to the cost drivers will be summed and converted into rating levels using the provided tables. The cost-drivers are grouped into four categories: Product, Platform, Personnel, and Project.

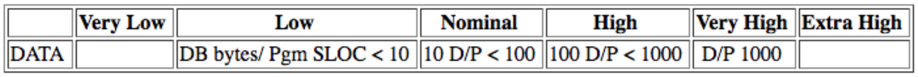
3.3.1 Product factors

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. We can have high financial losses due to software failures, so the value will be set to HIGH.



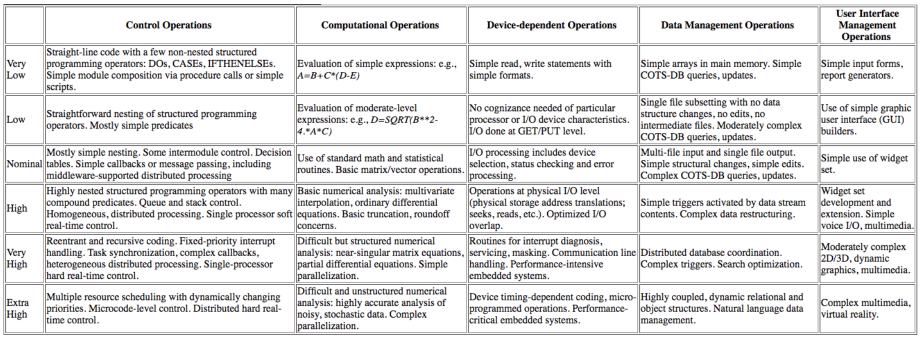
	Very Low	Low	Nominal	High	Very High	Extra High
RELY	slight inconvenience	low, easily recoverable losses	moderate, easily recoverable losses	high financial loss	risk to human life	

Data Base Size (DATA) : We will have an high number of reservations, rides and payments but each one of them need only a little amount of space. Therefore this value will be set to NOMINAL.



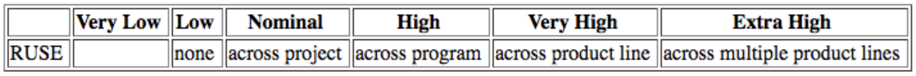
	Very Low	Low	Nominal	High	Very High	Extra High
DATA		DB bytes/ Pgm SLOC < 10	10 D/P < 100	100 D/P < 1000	D/P 1000	

Product Complexity (CPLX) : We have evaluated control operations, computational operations, device-dependent operations, data management operations, and user interface management operations' main characteristics and performances. So, accordingly with the COCOMO® II CPLX rating scale the value will be set to HIGH.



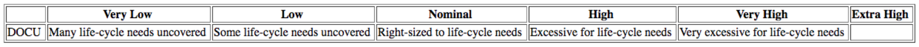
	Control Operations	Computational Operations	Device-dependent Operations	Data Management Operations	User Interface Management Operations
Very Low	Straight-line code with a few non-nested structured programming operators: DOs, CASEs, IFTHENELSEs. Simple module composition via procedure calls or simple scripts.	Evaluation of simple expressions: e.g., A=B+C*(D-E)	Simple read, write statements with simple formats.	Simple arrays in main memory. Simple COTS-DB queries, updates.	Simple input forms, report generators.
Low	Straightforward nesting of structured programming operators. Mostly simple predicates	Evaluation of moderate-level expressions: e.g., D=SQRT(B**2-4.*A*C)	No cognizance needed of particular processor or I/O device characteristics. I/O done at GET/PUT level.	Single file subsetting with no data structure changes, no edits, no intermediate files. Moderately complex COTS-DB queries, updates.	Use of simple graphic user interface (GUI) builders.
Nominal	Mostly simple nesting. Some intermodule control. Decision tables. Simple callbacks or message passing, including middleware-supported distributed processing	Use of standard math and statistical routines. Basic matrix/vector operations.	I/O processing includes device selection, status checking and error processing.	Multi-file input and single file output. Simple structural changes, simple edits. Complex COTS-DB queries, updates.	Simple use of widget set.
High	Highly nested structured programming operators with many compound predicates. Queue and stack control. Homogeneous, distributed processing. Single processor soft real-time control.	Basic numerical analysis: multivariate interpolation, ordinary differential equations. Basic truncation, roundoff concerns.	Operations at physical I/O level (physical storage address translations; seeks, reads, etc.). Optimized I/O overlap.	Simple triggers activated by data stream contents. Complex data restructuring.	Widget set development and extension. Simple voice I/O, multimedia.
Very High	Reentrant and recursive coding. Fixed-priority interrupt handling. Task synchronization, complex callbacks, heterogeneous distributed processing. Single-processor hard real-time control.	Difficult but structured numerical analysis: near-singular matrix equations, partial differential equations. Simple parallelization.	Routines for interrupt diagnosis, servicing, masking. Communication line handling. Performance-intensive embedded systems.	Distributed database coordination. Complex triggers. Search optimization.	Moderately complex 2D/3D, dynamic graphics, multimedia.
Extra High	Multiple resource scheduling with dynamically changing priorities. Microcode-level control. Distributed hard real-time control.	Difficult and unstructured numerical analysis: highly accurate analysis of noisy, stochastic data. Complex parallelization.	Device timing-dependent coding, micro-programmed operations. Performance-critical embedded systems.	Highly coupled, dynamic relational and object structures. Natural language data management.	Complex multimedia, virtual reality.

Developed for Reusability (RUSE) : Our efforts were aimed at the construction of components intended for reuse on the current or future projects. This effort is consumed with creating generic design of software, elaborate documentation, and extensive testing to ensure components are ready to be used in other applications. Therefore this value will be set to HIGH.



	Very Low	Low	Nominal	High	Very High	Extra High
RUSE		none	across project	across program	across product line	across multiple product lines

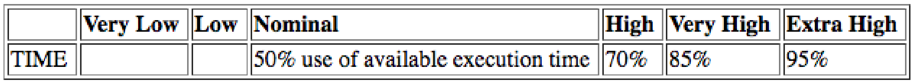
Documentation Match to Life-Cycle Needs (DOCU) : The project's documentation is suitable with its life-cycle needs. The value for this cost driver will be set to NOMINAL.



	Very Low	Low	Nominal	High	Very High	Extra High
DOCU	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	

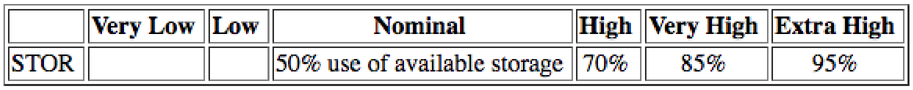
3.3.2 Platform factors

Execution Time Constraint (TIME) : The percentage of available-execution-time expected to be used by the system is quite low, less than 50% of the execution time resource is consumed. Therefore, the value will be set to NOMINAL.



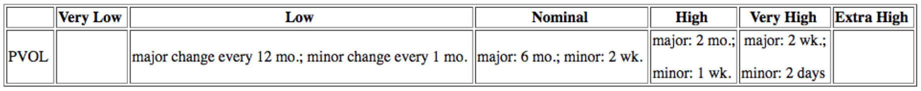
	Very Low	Low	Nominal	High	Very High	Extra High
TIME			50% use of available execution time	70%	85%	95%

Main Storage Constraint (STOR) : Our project continues to expand with a low rate, and to consume whatever resources are available, making available processor execution time and main storage cost drivers still relevant. The value will be set to NOMINAL.



	Very Low	Low	Nominal	High	Very High	Extra High
STOR			50% use of available storage	70%	85%	95%

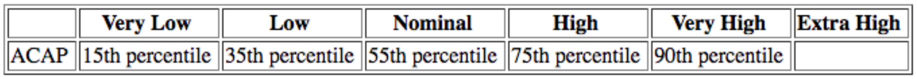
Platform Volatility (PVOL) : We suppose that in our system there can be a major change every 6 months and a minor change every 2 weeks. These are maybe due to changes in the mobile operation systems or for the release of new and more performant software that we can use in the system. So, the value will be set to NOMINAL.



	Very Low	Low	Nominal	High	Very High	Extra High
PVOL		major change every 12 mo.; minor change every 1 mo.	major: 6 mo.; minor: 2 wk.	major: 2 mo.; minor: 1 wk.	major: 2 wk.; minor: 2 days	

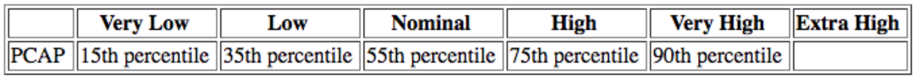
3.3.3 Personnel Factors

Analyst Capability (ACAP) : Our Analysis and Design ability, efficiency and thoroughness, and the ability to communicate and cooperate, fall in the 75th percentile. Then, the value will be set to HIGH.



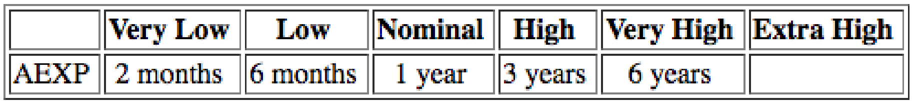
	Very Low	Low	Nominal	High	Very High	Extra High
ACAP	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	

Programmer Capability (PCAP) : E didn't programme our application so we chose a medium value for not compromise the calculation. The value will be set to NOMINAL.



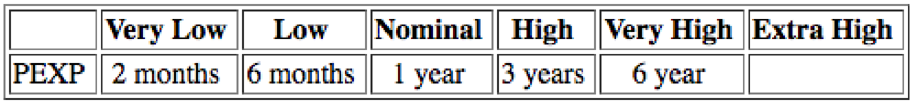
	Very Low	Low	Nominal	High	Very High	Extra High
PCAP	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	

Applications Experience (APEX) : Our level of experience with this type of application at the beginning of the project was very modest and then the value will be set to LOW.



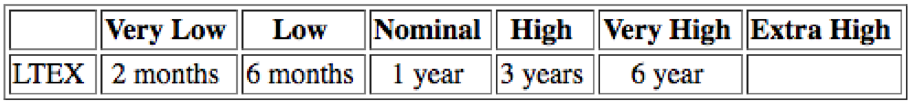
	Very Low	Low	Nominal	High	Very High	Extra High
AEXP	2 months	6 months	1 year	3 years	6 years	

Platform Experience (PLEX) : Our experience in recognize the importance of platforms, including more graphic user interface, database, networking, and distributed middleware capabilities is lower than six months. During other courses, we studied something useful here but most of the things were new for us and then the value will be set to LOW.



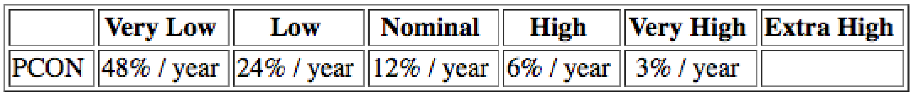
	Very Low	Low	Nominal	High	Very High	Extra High
PEXP	2 months	6 months	1 year	3 years	6 year	

Language and Tool Experience (LTEX) : We had little experience with Latex and Draw.io, the two most important tools for our project. The value will be set to NOMINAL.



	Very Low	Low	Nominal	High	Very High	Extra High
LTEX	2 months	6 months	1 year	3 years	6 year	

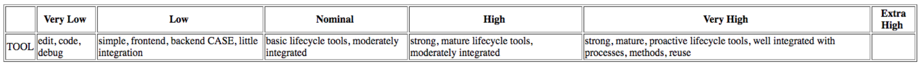
Personnel Continuity (PCON) : During the development of the project our team haven't got turnovers, so the value will be set to VERY HIGH.



	Very Low	Low	Nominal	High	Very High	Extra High
PCON	48% / year	24% / year	12% / year	6% / year	3% / year	

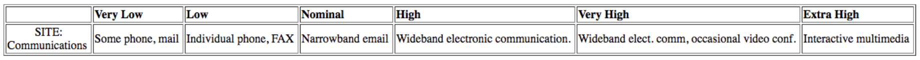
3.3.4 Project Factors

Use of Software Tools (TOOL) : We used strong, mature lifecycle tools, moderately integrated. Then, the value will be set to HIGH.



	Very Low	Low	Nominal	High	Very High	Extra High
TOOL	edit, code, debug	simple, frontend, backend CASE, little integration	basic lifecycle tools, moderately integrated	strong, mature lifecycle tools, moderately integrated	strong, mature, proactive lifecycle tools, well integrated with processes, methods, reuse	

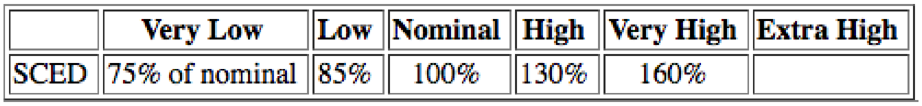
Multisite Development (SITE) : The project has an international distribution and has a full interactive multimedia access. The value will be set to VERY HIGH.



	Very Low	Low	Nominal	High	Very High	Extra High
SITE: Communications	Some phone, mail	Individual phone, FAX	Narrowband email	Wideband electronic communication.	Wideband elect. comm, occasional video conf.	Interactive multimedia

3.3.5 General Factor

Required Development Schedule (SCED) : Our schedule has been quite strict, the work had to be done on time so the stretch-out has been 100% and the value will be set to NOMINAL.



	Very Low	Low	Nominal	High	Very High	Extra High
SCED	75% of nominal	85%	100%	130%	160%	

Cost Driver	**Factor**	**Value**
RELY	high	1.15
DATA	nominal	1.08
CPLX	high	1.15
RUSE	high	1.15
DOCU	nominal	1.00
TIME	nominal	1.00
STOR	nominal	1.00
PVOL	nominal	1.00
ACAP	high	0.86
PCAP	nominal	1.00
APEX	low	1.10
PLEX	low	1.10
LTEX	nominal	1.00
PCON	very high	0.65
TOOL	high	0.91
SITE	very high	0.86
SCED	nominal	1.00

EAF	1.356

The EAF value is evaluated as:

$$\text{EAF} = \prod_{i=1}^{17} cost\_driver_i$$

3.4 Effort Equation

Thanks to the values obtained in the previous section, now we are able to evaluate an estimation of the efforts that are needed for the development of the project. Effort $= A * EAF * KSLOC^{E}$, where:

- A = 2.94

- In the FP we estimated the Size of the project with a lower-bound of 8.878 and an upper-bound of 12.931
- E = 1.0518 derived from the Scale drivers
- EAF = 1.356 derived from the Cost drivers

It is measured in Person-Months (PM). With the computed lower-bound we have:

$$Effort_l = 2.94 * 1.356 * 8.878^{1.052} \cong \mathbf{40\ PM}$$

with the upper:

$$Effort_u = 2.94 * 1.356 * 12.931^{1.052} \cong \mathbf{59\ PM}$$

In our Organization, 1PM = 152PH , so 8968 PH are estimated for the entire project with the higher Effort.

3.5 Schedule Estimation

To give an estimation of the duration of the project we can use the following formula:

$$\textbf{Duration} = 3.67 * \textit{Effort}^F$$

Where:

- F $= 0.28 + 0.2(E - B) \cong 0.308$
- $Effort_l = 40$ PM with the computed lower-bound, $Effort_u = 59$ PM with the upper-bound.

Then, with the lower-bound we have:

$$Duration_l \cong \mathbf{11.5\ months}$$

with the upper:

$$Duration_u \cong \mathbf{13\ months}$$

The duration estimated here is reasonable for this kind of project.

Finally we can estimate the number of team's components needed to complete the project:

$$Members_l = \frac{Effort}{Duration} \cong \mathbf{3}$$

in the lower-bound case, while with the upper-bound:

$$Members_u = \frac{Effort}{Duration} \cong \mathbf{5}$$

Generally, it's suggested to consider more the worst case then the best.

Schedule Planning and Resource Allocation

4.1 Identify the Deliveries

In our project, we have to release four documents, moreover we have to do a code inspection of an existing well--known open source project. So, we can naturally split our project in five deliveries:

- RASD: Requirements Analysis and Specification Document
- DD: Design Document
- ITPD: Integrated Test Planning Document
- PP: Project Plan
- Code inspection and bug identification activity.

For each delivery, we have to respect a submission deadline, reported in the following table.

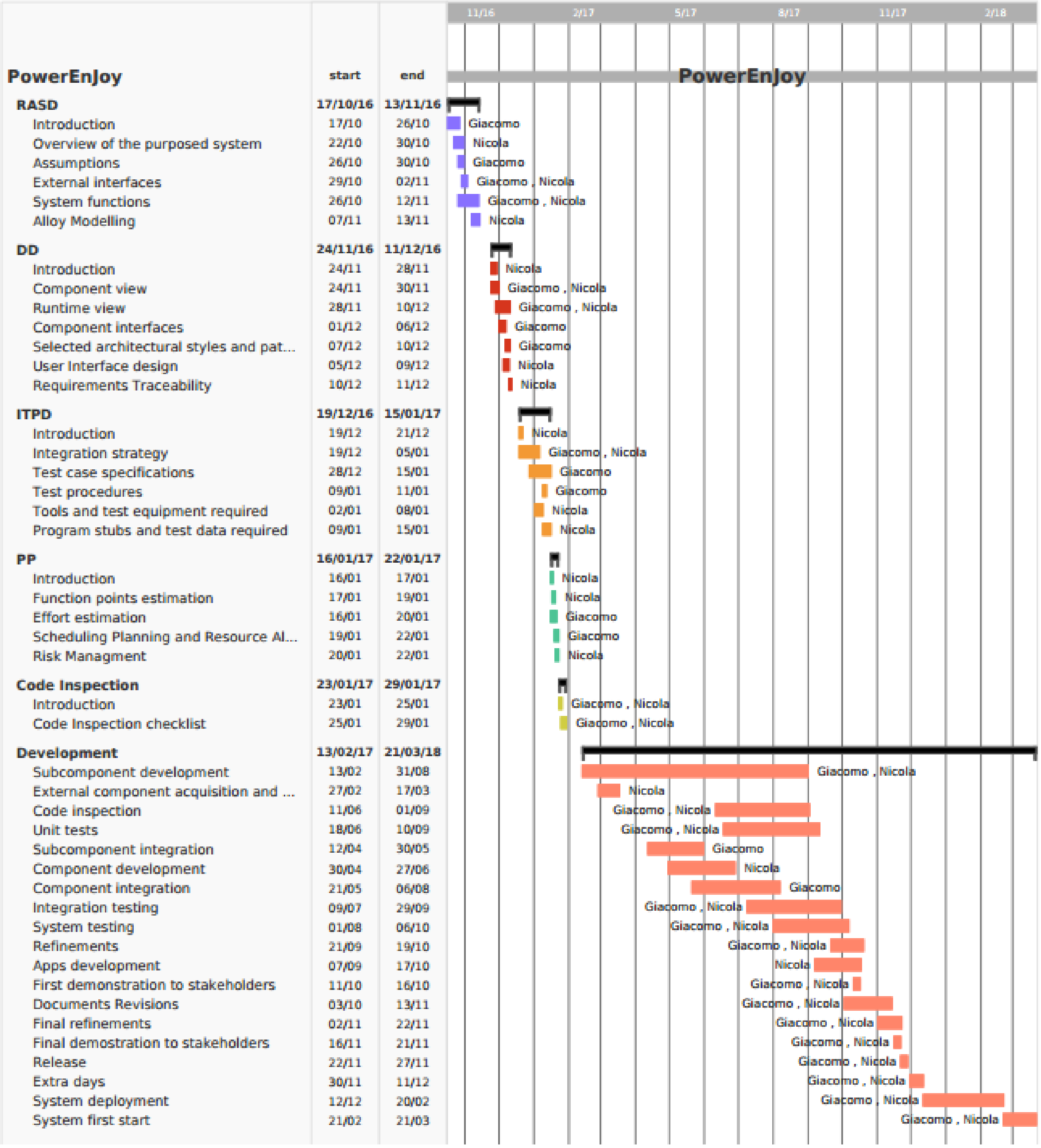
DELIVERY	**Submission deadline**
RASD	13/11/2016
DD	11/12/2016
ITPD	15/01/2017
PP	22/01/2017
Code inspecion	05/02/2017

Unfortunately, we can organize deliveries concurrently only in few cases. The RASD have to be completed before all the others, then the DD can be partially being done concurrently with the ITPD. The PP usually is done at the beginning of a project; in our case, it's done after the previous deliveries. The code inspection is executed on another project, so it can be done concurrently with every other delivery.

We decided to work on a delivery at a time in order to put all our efforts and reach the best possible result.

4.2 Scheduling

In this chapter, we are going to provide the PowerEnJoy schedule. It takes account only of the main tasks that we have to complete for our specific project. Each task has also associated the name of the component of the team that released it, or both if we did the task together.
The task during the project were split but here for a better understanding we reported only the main ones. We can also see that we frequently organized task concurrently.



PowerEnJoy	start	end
RASD	**17/10/16**	**13/11/16**
Introduction	17/10	26/10
Overview of the purposed system	22/10	30/10
Assumptions	26/10	30/10
External interfaces	29/10	02/11
System functions	26/10	12/11
Alloy Modelling	07/11	13/11
DD	**24/11/16**	**11/12/16**
Introduction	24/11	28/11
Component view	24/11	30/11
Runtime view	28/11	10/12
Component interfaces	01/12	06/12
Selected architectural styles and pat...	07/12	10/12
User Interface design	05/12	09/12
Requirements Traceability	10/12	11/12
ITPD	**19/12/16**	**15/01/17**
Introduction	19/12	21/12
Integration strategy	19/12	05/01
Test case specifications	28/12	15/01
Test procedures	09/01	11/01
Tools and test equipment required	02/01	08/01
Program stubs and test data required	09/01	15/01
PP	**16/01/17**	**22/01/17**
Introduction	16/01	17/01
Function points estimation	17/01	19/01
Effort estimation	16/01	20/01
Scheduling Planning and Resource Al...	19/01	22/01
Risk Managment	20/01	22/01
Code Inspection	**23/01/17**	**29/01/17**
Introduction	23/01	25/01
Code Inspection checklist	25/01	29/01
Development	**13/02/17**	**21/03/18**
Subcomponent development	13/02	31/08
External component acquisition and ...	27/02	17/03
Code inspection	11/06	01/09
Unit tests	18/06	10/09
Subcomponent integration	12/04	30/05
Component development	30/04	27/06
Component integration	21/05	06/08
Integration testing	09/07	29/09
System testing	01/08	06/10
Refinements	21/09	19/10
Apps development	07/09	17/10
First demonstration to stakeholders	11/10	16/10
Documents Revisions	03/10	13/11
Final refinements	02/11	22/11
Final demostration to stakeholders	16/11	21/11
Release	22/11	27/11
Extra days	30/11	11/12
System deployment	12/12	20/02
System first start	21/02	21/03

Risk Management

In the following sections the main identified risks for the PowerEnJoy development process will be classified and analyzed.

5.1 Project Risks

Milestone Delay : One of the identified internal risks is the possibility of delayed deadlines, it can in fact occur that one or more deadlines are missed because of seasonal illness, injuries or unexpected complications. A possible solution for this problem is to take into account some extra time and distribute the responsibilities equally among the team members so that no task would be seriously compromised by personal and temporary problems.

Knowledge Overestimation : Another internal risk might be the over-estimation of the knowledge in a specific field. A possible solution could be consulting an expert in the early phases of development.

Code Loss : Losing (a part of) the entire application source code would be an important loss for our team. This must be avoided but fortunately it is quite easy to deal with it: a possible solution is the adoption of an opportune distributed version-control system.

Lack of Communication : One of the main risks in a team-project is the lack of communication, this can lead to arguments and misunderstandings, thus heavily slowing the development. A possible solution would be a fair division of the responsibilities and the planning of frequent team-meeting to discuss the most critical points.

Risk	Probability	Impact
Milestone Delay	Average	Negligible
Knowledge Overestimation	Average	Critical
Code Loss	Low	Catastrophic
Lack of Communication	Average	Marginal

5.2 Technical Risks

External Services : Since our system relies heavily on external services like the Notification, Payment and Map ones, a slight modification in the usage terms or the deprecation of one or more API functionalities can lead to malfunctions and financial consequences. In order to avoid this kind of situations we need to design the code to be as modular as possible and constantly monitor our external services looking for updates.

Hardware Solutions : The PowerEnJoy system is not only a software system. Some of the analyzed risks are in fact typical for this type of application but, in particular, when an hardware-based project is developed one massive risk probably regards the hardware parts. The vehicles are subject to different kind of faults (sensors, engine, mechanical, software etc.) and this must be taken into account. This has partially been discussed in the previous documents, an OUTOFSERVICE status has been implemented but the vehicles need to be actually fixed and for this reason the team must contact the local government and local vehicle-reparation laboratories in order to find a suitable agreement.

Risk	Probability	Effects
External Services	Low	Critical
Hardware Solutions	High	Critical

5.3 Business Risks

Competitors Companies : One first risk for this category is the possibility of having competitors companies (eg. car-sharing companies, futuristic self-driving taxies) opposing to our business, for this reason the PowerEnJoy system must always provide competitive prices and functionalities.

Intended Users : Probably the most severe issues regards the acceptance of the system from its intended users. Since this service is eco-friendly and has already some effective discount plans, we can assume that it will insert quite easily in the current car-sharing ecosystem, but the development team should take into account future marketing strategies and appealing discount plans.

City Administration : The local government has a considereable influence on the PowerEnJoy system, in fact the entire system relies on stable agreements for circulation (Limited Traffic Zones) and access to public parkings. A possible solution would be the planning of periodical meetings to during the 'insediation' in order to define the roles of both our system and the local government and negotiate on the most critical points.

National Legislation : This is a critical problem for a system like PowerEnJoy: the car-sharing leglislation (eg. car insurance) can possibly change and our system, being phisically involved, needs to be always up-to-date. A possible solution is to actively watch out for these laws and, since they typically take months to be approved, and plan the future evolution of the system along with the entire team.

Risk	Probability	Effects
Competitors Companies	High	Marginal
Intended Users	High	Critical
City Administration	Low	Marginal
National Legislation	Low	Marginal

Appendix A: Used Tools

A.1 LaTeX

Used to format and redact this document

A.2 *g*it

Used as version control system in order to lead development

A.3 *t*eamgantt.com

Used for the Gantt chart

Appendix B: Hours of work

These are the hours of work spent by each group member in order to redact this document:

- Ruaro Nicola: 15 hours
- Gregori Giacomo: 15 hours
- Total worktime: 30 hours

Appendix C: Revisions

These sections will be eventually redacted during future post-release updates in order to approach the ITPD modifiability providing a comfortable and higly effective way to trace changes:

Acronyms

COCOMO Constructive Cost Model.

DB Database.

DBMS Database management system.

DD Design Document.

EAF Effort Adjustment Factor.

EI External Input.

EIS Enterprise Information System.

EJB Enterprise Java Beans.

ELF External logic file.

EO External Output.

EQ External Inquiries.

ER Entity-Relationship Diagram: diagram that shows the relationships of entity sets stored in a database.

FP Function Points.

ILF Internal logic file.

ITPD Integration Test Plan Document.

JEB Java Entity Bean.

PP Project Plan.

RASD Requirements Analysis and Specification Document.

RDBMS Relational database management system.

SQL Structured Query Language.

Bibliography

[1] Luca Mottola and Elisabetta Di Nitto, *Software Engineering 2: Project goal, schedule and rules*, 2016

[2] Nicola Ruaro and Giacomo Gregori, *RASD: Requirements Analysis and Specification Document*, 2016

[3] Nicola Ruaro and Giacomo Gregori, *DD: Design Document*, 2016

[4] Nicola Ruaro and Giacomo Gregori, *ITPD: Integrated Test Planning Document*, 2017

[5] Center for Software Engineering, USC, *The COCOMO II Model Definition Manual* , 1995 - 2000