POLITECNICO
MILANO 1863

PowerEnJoy

Project Plan Document

Luca Scannapieco - 877145
Andrea Pasquali - 808733
Emanuele Torelli - 876210

22/01/2017

Table of contents

1. Introduction

1.1. Purpose and scope

The aim of the Project Plan Document for PowerEnJoy is to estimate the complexity of the project in terms of size, cost and effort, in order to use this analysis subsequently as a guidance to define the required budget, the resources allocation and the schedule of the activities. The structure of the document is as following:

- In the first part, we use the function points approach to evaluate the size of the project.
- Then, with COCOMO II techniques, we evaluate the expected effort to be spent in the project.
- Once done these analysis, we can proceed with the allocation of the resources and the schedule of the activities.
- At last, we try to outline some of the risk that the PowerEnJoy project may encounter, giving possible solutions and suggestions.

1.2. List of definitions and abbreviations

- RASD: the Requirement Analysis and Specification Document provided before
- DD: the Design Document provided before
- ITPD: the Integration Test Plan Document provided before
- FP: function points (see section 2. for details)
- ILF: internal logic file, EIF: external interface file, EI: external input, EO: external output, EQ: external inquiries (see section 2. for details)
- DET: data element type, RET: record element type, FTR: file type referenced (see section 2. for details)
- SLOC: source lines of code
- COCOMO: approach used to get an estimation of effort invested in a project
- RELY, DATA, CPLX, RUSE, DOCU: COCOMO scale factors (see section 3.3. for details)
- TIME, STOR, PVOL, ACAP, PCAP, APEX, PLEX, LTEX PCON, TOOL, SITE, SCED: COCOMO cost drivers (see section 3.4. for details)

2. Size estimation: function points (FP)

The function points approach has the aim to provide an estimation of the size of the PowerEnJoy system, based on the complexity of the functionalities the system has to accomplish and the data structures involved.

First of all, it's fundamental to provide some definitions to clarify the key elements of the analysis. To evaluate the size of the project, the rating is based upon the number of data element types (DET), the record element types (RET) and the file types referenced (FTR).

- DET: data element type, user recognizable subgroup of data elements within an ILF or an EIF.
- RET: record element type, a file type referenced by a transaction. An FTR must also be an internal logical file or external interface file.
- FTR: file type referenced, a unique user recognizable, non-repetitive field.

We are going to identify such RETs, DETs and FTRs into five different kinds of function types:

- ILF: internal logic file, homogeneous set of data used and managed by the application.
- EIF: external interface file, homogeneous set of data used by the application but generated and maintained by other applications.
- EI: external input, elementary operation to elaborate data coming from the external environment.
- EO: external output, elementary operation that generates data for the external environment.
- EQ: external inquiries, elementary operation that involves input and output without significant elaboration of data from logic files.

Once clarified the meaning of these acronyms, we provide a table regarding the analysis of complexity of the internal logic files and external interface files (ILFs and EIFs), based on the combined quantity of DETs and RETs:

	DET		
RET	1-19	20-50	>50
1	Low	Low	Average
2-5	Low	Average	High
>5	Average	High	High

Then we show the table concerning the analysis of the complexity of external inputs (EIs), based on the combined quantity of DETs and RETs:

	DET		
FTR	1-4	5-15	>15
0-1	Low	Low	Average
2	Low	Average	High
>2	Average	High	High

At last, this is the table concerning the analysis of the complexity of external outputs (EOs) and external inquiries (EQs), based on the combined quantity of DETs and RETs:

	DET		
FTR	1-5	6-19	>19
0-1	Low	Low	Average
2-3	Low	Average	High
>3	Average	High	High

Once assessed the complexity of every function type, we can proceed quantifying the FPs for each element according to the different complexities, as shown in the following table:

Function types	**Complexity weight**		
	Low	Average	High
Internal logic files (ILF)	7	10	15
External interface files (EIF)	5	7	10
External inputs (EI)	3	4	6
External outputs (EO)	4	5	7
Eternal inquiries (EQ)	3	4	6

2.1. Internal logic files (ILF)

In this section we are going to find out all the possible ILFs involved in the processes of PowerEnJoy system, and we will indicate for each ILF the estimated complexity according to the number of DETs and RETs, as showed in the previous table.

- The system has to manage the information about the registered users, which include name, surname, phone number, email, SSN, credit card number and driving licence number, together with a password provided by the system during the registration phase and the always available user's position. Even if may seem that there are a lot of information about a registered user, the DETs are only 9, and moreover they all can be stored in a single table or a flat data structure involving a single RET. The complexity is low and the FPs produced are 7.
- The system has also the necessity to collect information about cars, which include the ID number, the position, the battery level, a Boolean to indicate if it's in charge, and the state (available, unavailable or out of order). Also in this case all the elements can be represented by a flat data structure, so the RET is one and there are 5 DETs. The complexity is therefore low and the FPs produced are 7.
- A reservation is made of a user and a car, which refers to other two records, together with a reservation timer. The RETs are 3 and the DETs are 1+9+5=15, so the complexity is again low and the FPs produced are 7.
- Safe areas consist on a set of positions, and have a set of power stations, so because of the large amount of data and records they can contain, the complexity is high and the FPs produced are 15.
- A power station has its own position and a Boolean to indicate the availability. There is only a RET and 2 DETs, so the complexity is low and the FPs produced are 7.
- The money saving option contains a starting position, an ending position and the best power station to leave the car, which produces one more record with 2 data elements. The total RETs are 2 and the DETs are 4, so the complexity is low and the FPs produced are 7.
- A more complex situation is the management of a ride, because it depends on plenty information, which can be primitive data like duration, the position, number of passenger, total price, battery level at the end of the ride itself, some Booleans that indicate the presence of the money saving option, the termination of the ride, if the car is left in charge at the end of the ride, and accidents (9 DETs in total), as well as other data objects like reservation (3 additional RETs with 15 total DETs), information about the money saving option (2 additional RETs with 4 total DETs), and the car (1 additional RETs with 5 total DETs). The total number of RETs for the ride is 1+3+2+1=6, and the number of DETs is 9+15+4+5=33. The complexity is high and the FPs produced are 15.

The following table contains a recap of the analysis of the complexity of the various ILFs and the FPs derived from each of them, including the total number of FPs produced.

ILF	**Complexity**	**FPs**
User	Low	7
Car	Low	7
Reservation	Low	7
Safe area	High	15
Money saving option	Low	7
Power station	Low	7
Ride	High	15
Total		65

2.2. External interface files (EIF)

PowerEnJoy system doesn't rely on many EIFs, since it doesn't cooperate with many external services. The only feature it must accomplish is to render maps on the client side applications. Since the interaction is quite complex and there is a big amount of data retrieved, it's reasonable to think that such EIFs are composed of a high number of data elements and records, so they implicate a high complexity.

The following table contains a recap of the analysis of the complexity of the various ILFs and the FPs derived from each of them, including the total number of FPs produced.

EIF	**Complexity**	**FPs**
Mapping service on client app	High	10
Mapping service on car app	High	10
Mapping service on assistance coordinator program	High	10
Total		30

2.3. External inputs (EI)

PowerEnJoy supports many interactions with users among different interfaces. In this section we are going to identify all the main functionalities offered by the system with the corresponding complexity in terms of EIs.

On the client app (mobile app or web app):

- Sign up: this operation involves all the data provided by the user, which contains 9 DETs, and, to ensure the validity of the fields and store data successfully, creates the ILF related to the user, hence the complexity is low and the FPs to consider are 3.
- Login: this is a simpler operation than the sign up one because involves less DETs, therefore the FPs are 3.
- Make a reservation: this operation involves the creation of the ILF Reservation associated to the corresponding user and car (other two FTRs), but the number of DETs involved isn't much large, so the complexity is average and it contributes 5 FPs.
- Unlock a car: this operation is quite complex, because it needs to check if the user is trying to unlock the car he had reserved, and it has also to check if he is less distant than 5m from the car. The operation involves many FTRs as well as DETs, so the FPs contributed are 6.

On the car app:

- Start a ride: this operation complex since it creates the object Ride, which requires many DETs, associated to the right user, so it yields 6 FPs.
- Activate the money saving option: this is probably the most complex inquiry, because the system has to display, according to the destination provided by the user and the distribution of the cars within the city, the best power station where the user can leave the car. This operation involves a huge amount of data, so it contributes 6 FPs.
- Finish a ride: this operation involves a flow of information between the server and the car, since the server has to check if the user is attempting to park the car in a safe area (or the sensors detected that an accident occurred). The complexity is average, so the FPs produced are 4.
- Plug the car into a power station: once a user terminates his ride, he has 3 minutes to eventually plug the car into a power station in order to get a discount. This operation is quite simple, since the system has only to check in the FTR related to the ride if the timer expires, so it contributes 3 FPs.

On the assistance coordinator program:

- Login: as the counterpart used to authenticate the access of users, this is a simple operation and yields 3 FPs.
- Tag/untag a car as out of order: this is a simple operation which only changes an attribute of the selected car, therefore it produces 3 FPs.

The following table contains a recap of the analysis of the complexity of the various EIs and the FPs derived from each of them, including the total number of FPs produced.

EI	**Complexity**	**FPs**
Sign up	Low	3
Login (user)	Low	3
Make a reservation	Average	4
Unlock a car	High	6
Start a ride	High	6
Activate the money saving option	High	6
Finish a ride	Average	4
Plug the car into a power station	Low	3
Login (assistance coordinator)	Low	3
Tag/untag a car as out of order	Low	3
Total		41

2.4. External outputs (EO)

As a part of its normal functionalities, the PowerEnJoy system occasionally needs to communicate with the user outside the context of an inquiry. These occasions are:

- Provide a password to the user (end of sign up process): this operation may be seen as a part of the signing up procedure, but due to a significant elaboration of logic files, we decided to split them up and to indicate the giving of the password to the user as an external output procedure. This is not a complex procedure because doesn't rely on many FTRs and DETs, so it yields 4 FPs.
- See all the information about a ride: during a ride, the car app is supposed to show the necessary information about a ride, like cost and duration, as well as a map with safe areas and power stations. Since this procedure involves a quite massive retrieval of data, we consider this operation as complex and the FPs produced are 7.
- Show the final cost of the ride: this is a complex operation since the system has to check a large number of parameters including the car's battery, the car's position, the number of passengers during a ride, if the car is left in charge and so on. All of these operations are in a strict sequence due to the priorities that the various discounts hold. Therefore the complexity is very high and the contribute is 7 FPs.
- Apply the 1 Euro fine if the reservation expires: this operation is simple because it only needs to check when a reservation gets out of the allowed time, so it involves only the related FTR and few DETs. The contribute is 4 FPs.

The following table contains a recap of the analysis of the complexity of the various EOs and the FPs derived from each of them, including the total number of FPs produced.

EO	Complexity	FPs
Provide a password to the user (end of sign up process)	Low	4
See all the information about a ride	High	7
Show the final cost of the ride	High	7
Apply the 1 Euro fine if the reservation expires	Low	4
Total		22

2.5. External inquiries (EQ)

An inquiry is a data retrieval request performed by a user.

- See available cars (user – current position): the user clicks a button and then he's allowed to see all the cars in a map, so it is a trivial operation and contributes 3 FPs.
- See available cars (user – given address): almost the same operation mentioned before, with the difference that instead of retrieving the user's position, the address is given by the user himself. The FPs produced are 3.
- See position and battery of cars (assistance coordinators): it's essentially the same operation that a user can do, except from the fact that the assistance coordinator can see all the cars, therefore the contribution is again 3 FPs.
- Show battery and position of the reserved car: the system allows the owner of a reservation to see the position and the battery of the reserved car. This operation is simple, it produces 3 FPs.

The following table contains a recap of the analysis of the complexity of the various EQs and the FPs derived from each of them, including the total number of FPs produced.

EQ	Complexity	FPs
See available cars (user – current position)	Low	3
See available cars (user – given address)	Low	3
See position and battery of cars (assistance coordinators)	Low	3
Show battery and position of the reserved car	Low	3
Total		12

2.6. Overall estimation

The following table provides a summary of the previous analysis:

Function Type	Value
Internal logic files (ILF)	65
External interface files (EIF)	30
External inputs (EI)	41
External outputs (EO)	22
External inquiries (EQ)	12
Total	170

3. Cost and effort estimation: COCOMO II

In this section we will adopt the COCOMO approach in order to get an estimation of effort that is needed to spend to complete the PowerEnjoy project. This method is essentially based on the following basic formula that estimates the effort in Person-Month:

$$PM = A * Size^{E} * \prod_{i=1}^{n} EM_i$$

Where:

- A=2.94 is a COCOMO constant.
- Size is the result of the size estimation
- E is given from the following formula:

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

  where:
  B=0.91 is a COCOMO constant
  SF_j is the j-th scale factor (computed in the section 3.2.)
- EM_i is the i-th cost driver (computed in the section 3.3.)

In essence, we are going to model, as more accurately as possible many aspects of our project and compute, on this model, the value of the PM index.

3.1. Post-architecture or Early-design

Even if we are projecting a totally new artefact we are going to carry out our effort estimation following the post-architecture model. This choice is given from the fact that, at this stage of our work, we have already chosen a quite detailed architecture for our system to be. In fact we have already delivered the design document to our customers, so we think we have enough information about the architecture of our system to follow the post-architecture model.

3.2. Size parameter estimation

The Size parameter is the estimated size of the project, expressed in KSLOC (Kilo-Source Lines of Code). It can be deducted from the UFP computed in the section 2.1 with the following formula:

$$SIZE = AVC * \#FP$$

i.e. it's the product of the number of functional points and a parameter that expresses the average number of LOC per FP for a given programming language. Since we are going to build our project with JEE this parameter is (source: Quantitative Software Management):

$$AVC = 46$$

Thus the average number of LOC of our project will be:

$$SIZE = AVC * \#FP = 46 * 169 = 7774\ LOC$$

3.3. Scale factors

The following official COCOMO II table shows the value of the five COCOMO scale factors for each rating level (from very low to extra high). We are going to choose a rating level for each scale factor.

Scale factor	Very Low	Low	Nominal	High	Very High	Extra High
PREC	Thoroughly unprecedented	Largely unprecedented	Somewhat unprecedented	Generally familiar	Largely familiar	Thoroughly familiar
SF_i	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	Rigorous	Occasional relaxation	Some relaxation	General conformity	Some conformity	General goals
SF_i	5.07	4.05	3.04	2.03	1.01	0.00
RESL	Little (20%)	Some (40%)	Often (60%)	Generally (75%)	Mostly (90%)	Full (100%)
SF_i	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	Very difficult interaction	Some difficult interactions	Basically cooperative	Largely cooperative	Highly cooperative	Seamless interactions
SF_i	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	SW-CMM Level 1	SW-CMM Level 1	SW-CMM Level 2	SW-CMM Level 3	SW-CMM Level 4	SW-CMM Level 5
SF_i	7.80	6.24	4.68	3.12	1.56	0.00

In the following five sections we are going to argue the choice of the rating level for each scale factor. These choices have been taken considering, for each scale factor, a criterion (illustrated as tables that can be found in the COCOMO model definition manual) that map some project features onto the scale factors.

3.3.1. Precedentedness (PREC)

It's high if the product is similar to several previously developed projects.

Feature	Very low	Nominal/High	Extra High
Organizational understanding of product	General	Considerable	Thorough
Experience in working with related software systems	Moderate	Considerable	Extensive
Concurrent development of associated new hardware and operational procedures	Extensive	Moderate	Some
Need for innovative data processing architectures, algorithms	Considerable	Some	Minimal

No particular relationship between developers and costumers leads to a general organizational understanding of the product. For all the members of our group this is the first large scale project development. The rating level is low.

3.3.2. Development flexibility (FLEX)

It reflects the degree of flexibility in the development process with respect to the external specification and requirements.

Feature	Very low	Nominal/High	Extra High
Need for software conformance with pre-established requirements	Full	Considerable	Basic
Need for software conformance with external interface specifications	Full	Considerable	Basic

High need of meeting the requirements but just basic needs of conformance with external interfaces. The rating level is high.

3.3.3. Risk Resolution (RESL)

It reflects the level of awareness and reactiveness with respect to risks.

Characteristic	Very Low	Low	Nominal	High	Very High	Extra High
Risk Management Plan identifies all critical risk items, establishes milestones for resolving them by PDR or LCA.	None	Little	Some	Generally	Mostly	Fully
Schedule, budget, and internal milestones through PDR or LCA compatible with Risk Management Plan.	None	Little	Some	Generally	Mostly	Fully
Percent of development schedule devoted to establishing architecture, given general product objectives.	5	10	17	25	33	40
Tool support available for resolving risk items, developing and verifying architectural specs	None	Little	Some	Good	Strong	Full
Level of uncertainty in key architecture drivers: mission, user interface, COTS, hardware, technology, performance.	Extreme	Significant	Considerable	Some	Little	Very little
Number and criticality of risk items.	>10 Critical	5-10 Critical	2-4 Critical	1 Critical	>5 Non-Critical	<5 Non-Critical

Since, at this stage, we have not started the implementation yet, we are very uncertain about the key architecture drivers because we think that we are going to revise some architectural choices during the implementation phase. On the other hand, we have tried to spend some effort in the identification of risks and their resolutions. The rating level is nominal.

3.3.4. Team Cohesion (TEAM)

It reflects the degree of difficulty in synchronizing the project's stakeholders: users, customers, developers, maintainers.

Characteristic	Very Low	Low	Nominal	High	Very High	Extra High
Consistency of stakeholder objectives and cultures	Little	Some	Basic	Considerable	Strong	Full
Ability, willingness of stakeholders to accommodate other stakeholders' objectives	Little	Some	Basic	Considerable	Strong	Full
Experience of stakeholders in operating as a team	None	Little	Little	Basic	Considerable	Extensive
Stakeholder teambuilding to achieve shared vision and commitments	None	Little	Little	Basic	Considerable	Extensive

The members of our team cooperate very well, but we must consider the inevitable divergences with other stakeholders such as users and costumers. The rating level is high.

3.3.5. Process Maturity (PMAT)

Refers to a well-known method for assessing the maturity of a software organization, CMM, now evolved into CMMI.

PMAT Rating	Maturity Level	Process Characteristics
Very low/low	CMM level 1 Initial	Process is unpredictable, poorly controlled, and reactive
Nominal	CMM level 2 Managed	Process is characterized for projects and is often reactive
High	CMM level 3 Defined	Process is characterized for the organization and is proactive
Very high	CMM level 4 Quantitatively managed	Process is measured and controlled

Extra high	CMM level 5 Optimizing	Focus is on continuous quantitative improvement

We think our project is not so advanced to contain proactive processes. It's often reactive and relies on the knowledge of the three group members and their good interaction. The rating level is nominal.

3.5. Scale factors overview

The following table summarizes the choices of the scale factor analysis.

Scale factor	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 interaction 5.48	Some difficult interactions 4.38	Basically cooperative 3.29	Largely cooperative 2.19	Highly cooperative 1.10	Seamless interactions 0.00
PMAT SF_i	SW-CMM Level 1 7.80	SW-CMM Level 1 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

According to these values the exponent E will be:

$$E = B + 0.01 * \sum_{i=1}^{5} SF_i = 0.91 + 0.01 * (4.96 + 2.03 + 4.24 + 2.19 + 4.68) = 1.091$$

3.6. Cost drivers and effort multipliers

Since we have chosen the Post-Architecture model we have 17 cost drivers grouped in four categories. In the following sections we are going to illustrate, through COCOMO tables, all the rating level we have chosen for each cost driver.

3.6.1. 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.

RELY descriptors	Slightly inconvenience	Easily recoverable losses	Moderate recoverable losses	High financial loss	Risk to human life	
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort Multiplier (EM)	0.82	0.92	1.00	1.10	1.26	n/a

Hypothetic faults of PowerEnjoy application are going neither to cause high financial losses (as a bank application can do) nor to endanger human lives (as airplane software can do). We think system bugs can cause moderate recoverable losses such as not payed rides. Rating level is nominal.

3.6.2. Database size (DATA)

This cost driver attempts to capture the effect large test data requirements have on product development. The rating is determined by calculating D/P, the ratio of bytes in the testing database to SLOC in the program.

DATA descriptors		$\frac{Testing\ DB\ bytes}{Program\ SLOC} < 10$	$10 \leq \frac{D}{P} \leq 100$	$100 \leq \frac{D}{P} \leq 1000$	$\frac{D}{P} > 1000$	
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort Multiplier (EM)	n/a	0.90	1.00	1.14	1.28	n/a

We don't have the effective size of our database, but we estimate that it will at least 1GB. Since our SLOC is relatively low (see size estimation), we end up with the following lower bound for the ratio:

$$\frac{Testing\ DB\ bytes}{Program\ SLOC} = \frac{10^9\ byte}{7774\ SLOC} = 128634 * 10^6$$

The rating level is very high.

3.6.3. Product complexity (CPLX)

Complexity is divided into five areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations. Using the following table, we select the area or combination of areas that characterize the product or the component of the product. The complexity rating is the average of the selected area ratings.

	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, IFTHEN-ELSEs. Simple module composition via	Evaluation of simple expressions: e.g., A=B+C*(DE)	Simple read, write statements with simple formats.	Simple arrays in main memory. Simple COTSDB queries, updates.	Simple input forms, report generators.

	procedure calls or simple scripts.				
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 COTSDB 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, round-off 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	Re-entrant 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, microprogrammed operations. Performance-critical embedded systems.	Highly coupled, dynamic relational and object structures. Natural language data management.	Complex multimedia, virtual reality, natural language interface.

The average of the five areas ratings is nominal.

Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort Multiplier (EM)	0.73	0.87	1.00	1.17	1.34	1.74

3.6.4. Required reusability (RUSE)

This cost driver accounts for the additional effort needed to construct components intended for reuse on current or future projects. This effort is consumed with creating more generic design of software, more elaborate documentation, and more extensive testing to ensure components are ready for use in other applications. “Across project” could apply to reuse across the modules in a single financial applications project. “Across program” could apply to reuse across multiple financial applications projects for a single organization. “Across product line” could apply if the reuse is extended across multiple organizations. “Across multiple product lines” could apply to reuse across financial, sales, and marketing product lines.

RUSE descriptors		None	Across project	Across program	Across product line	Across multiple product lines
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort Multiplier (EM)	n/a	0.95	1.00	1.07	1.15	1.24

Since we are not an organization and we are not supposed to project other applications, we want the components of our project to be reusable just for the current project. The rating level is nominal.

3.6.5. Documentation match to life-cycle needs (DOCU)

The rating scale for the DOCU cost driver is evaluated in terms of the suitability of the project’s documentation to its life-cycle needs. The rating scale goes from very low (many life-cycle needs uncovered) to very high (very excessive for life-cycle needs).

DOCU descriptors	Many life-cycle needs uncovered	Some life-cycle needs uncovered	Right-sized to life-cycle needs	Excessive for life-cycle needs	Very excessive for life-cycle needs	
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort Multiplier (EM)	0.81	0.91	1.00	1.11	1.23	n/a

We are spending many efforts in documentation and we will probably end up with an excessive documentation for life-cycle needs. The rating level is high.

3.6.6. Execution time constraint (TIME)

This is a measure of the execution time constraint imposed upon a software system. The rating is expressed in terms of the percentage of available execution time expected to be used by the system consuming the execution time resource. The rating ranges from nominal, less than 50% of the execution time resource used, to extra high, 95% of the execution time resource is consumed.

TIME descriptors			≤ 50% use of available	70% use of available	85% use of available	95% use of available

			execution time	execution time	execution time	execution time
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort Multiplier (EM)	n/a	n/a	1.00	1.11	1.29	1.63

PowerEnjoy system has a couple of main functionalities (especially those regarding the location of cars and users) that will require a consistent CPU usage. The rating level is high.

3.6.7. Storage constraint (STOR)

This rating represents the degree of main storage constraint imposed on a software system. The rating ranges from nominal (less than 50%), to extra high (95%).

STOR descriptors			≤ 50% use of available storage	70% use of available storage	85% use of available storage	95% use of available storage
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort Multiplier (EM)	n/a	n/a	1.00	1.05	1.17	1.46

Since common hard drives can easily contain several Terabytes of data, we think this constraint variable is irrelevant for the size of our project. The rating level is nominal.

3.6.8. Platform volatility (PVOL)

"Platform" is used here to mean the complex of hardware and software (OS, DBMS, etc.) the software product calls on to perform its tasks. The platform includes any compilers or assemblers supporting the development of the software system. This rating ranges from low, where there is a major change every 12 months, to very high, where there is a major change every two weeks.

RELY descriptors		Major change every 12 month, minor-ones every 1 month	Major change every 6 month, minor-ones every 2 weeks	Major change every 2 month, minor-ones every 1 week	Major change every 2 weeks, minor-ones every 2 days	
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort Multiplier (EM)	n/a	0.87	1.00	1.15	1.30	n/a

The main platforms of our software product are mobile phone OSes and browsers. A mobile OS is subject to a major upgrade once per year, and a couple of minor upgrades per month; a browser is even less upgraded than mobile OSes. The rating level is LOW.

3.6.9. Analyst Capability (ACAP)

Analysts are personnel who work on requirements, high-level design and detailed design. The major attributes that should be considered in this rating are analysis and design ability, efficiency and thoroughness, and the ability to communicate and cooperate. The rating should not consider the level of experience of the analyst; that is rated with APEX, LTEX, and PLEX. Analyst teams that fall in the fifteenth percentile are rated very low and those that fall in the ninetieth percentile are rated as very high.

RELY descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort Multiplier (EM)	1.42	1.19	1.00	0.85	0.71	n/a

In our case the analysts and who are rating them are the same people. Since we trust in our work, we think we have carried out a very precise and thorough requirement analysis in the RASD and a well-done design in the DD. Rating level is very high.

3.6.10. Programmer capability (PCAP)

Evaluation should be based on the capability of the programmers as a team rather than as individuals. Major factors which should be considered in the rating are ability, efficiency and thoroughness, and the ability to communicate and cooperate. The experience of the programmer should not be considered here; it is rated with APEX, LTEX, and PLEX. A very low rated programmer team is in the fifteenth percentile and a very high rated programmer team is in the ninetieth percentile.

RELY descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort Multiplier (EM)	1.34	1.15	1.00	0.88	0.76	n/a

We are not going to implement the PowerEnJoy project. The rating level is nominal.

3.6.11. Application experience (APEX)

The rating for this cost driver is dependent on the level of applications experience of the project team developing the software system. The ratings are defined in terms of the project team's equivalent level of experience with this type of application. A very low rating is for application experience of less than 2 months. A very high rating is for experience of 6 years or more

RELY descriptors	≤ 2 months	6 months	1 years	3 years	6 years	
Rating level	Very low	Low	Nominal	High	Very high	Extra high

Effort Multiplier (EM)	1.22	1.10	1.00	0.88	0.81	n/a

For every member of our team this is his first experience in projecting a JEE system like PowerEnjoy. The rating level is very low.

3.6.12. Platform experience (PLEX)

The rating for this cost driver is dependent on the level of experience that the project team has with powerful platforms, including more graphic user interface, database, networking, and distributed middleware capabilities.

RELY descriptors	≤ 2 months	6 months	1 years	3 years	6 years	
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort Multiplier (EM)	1.19	1.09	1.00	0.91	0.85	n/a

No members of our team have previous experiences in projects with graphic user interfaces or the platforms above-mentioned. The rating level is very low.

3.6.13. Language and Tool experience (LTEX)

This is a measure of the level of programming language and software tool experience of the project team developing the software system. In addition to experience in the project's programming language, experience on the project's supporting tool set also affects development effort. A low rating is given for experience of less than 2 months. A very high rating is given for experience of 6 or more years.

RELY descriptors	≤ 2 months	6 months	1 years	3 years	6 years	
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort Multiplier (EM)	1.20	1.09	1.00	0.91	0.84	n/a

All the members of the team have 1 year of experience with Java programming language, but any experience with the major part of the tool used in the project (such as testing tools mentioned in the ITPD) and with JEE. The rating level is LOW.

3.6.14. Personnel continuity (PCON)

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

RELY descriptors	48% / year	24% / year	12% / year	6% / year	3% / year	

Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort Multiplier (EM)	1.29	1.12	1.00	0.90	0.81	n/a

We estimate a high level of continuity of the personnel, since the latter is composed of the three members of the group. The rating level is high.

3.6.15. Usage of software tools (TOOL)

The tool rating ranges from simple edit and code, very low, to integrated life-cycle management tools, very high.

RELY descriptors	Edit, code, debug	Simple, frontend, backend CASE, little integration	Basic life-cycle tools, moderately integrated	Strong, mature, proactive life-cycle tools, moderately integrated	Strong, mature, proactive life-cycle tools, well integrated with processes, methods, reuse	
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort Multiplier (EM)	1.17	1.09	1.00	0.90	0.78	n/a

Development environments like Eclipse or Netbeans are considered complete and well-integrated. They offer to the developers almost all support they need. The rating level is high.

3.6.16. Multisite development (SITE)

Determining the SITE cost driver rating involves the assessment and judgement-based averaging of two factors: site collocation (from fully collocated to international distribution) and communication support (from surface mail and some phone access to full interactive multimedia). For example, if a team is fully collocated, it doesn't need interactive multimedia to achieve an Extra High rating. Narrowband e-mail would usually be sufficient.

RELY descriptors	International Some phone, mail	Multi-city and multi-company Individual phone, fax	Multi-city or multi-company Narrow band email	Same city or metro area Wideband electronic communication	Same building or complex Wideband electronic communication, occasional video conference	Fully collocated Interactive multimedia
Rating level	Very low	Low	Nominal	High	Very high	Extra high

Effort Multiplier (EM)	1.22	1.09	1.00	0.93	0.86	0.80

Our team is fully collocated, in fact we live in the same city (and even in the same zone of the city). The rating level is extra high.

3.6.17. Required development schedule (SCED)

This rating measures the schedule constraint imposed on the project team developing the software. The ratings are defined in terms of the percentage of schedule stretch-out or acceleration with respect to a nominal schedule for a project requiring a given amount of effort. Accelerated schedules tend to produce more effort in the earlier phases to eliminate risks and refine the architecture, more effort in the later phases to accomplish more testing and documentation in parallel. In following table, schedule compression of 75% is rated very low. A schedule stretch-out of 160% is rated very high. Stretch-outs do not add or decrease effort. Their savings because of smaller team size are generally balanced by the need to carry project administrative functions over a longer period of time.

RELY descriptors	75% of nominal	85% of nominal	100% of nominal	130% of nominal	160% of nominal	
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort Multiplier (EM)	1.43	1.14	1.00	1.00	1.00	n/a

The schedule of our project will be significantly stretched-out because of the small size of the team with respect to the size of the project (if we imagine to implement PowerEnjoy application). The rating level is very high.

3.7. Cost driver overview

The following table shows the rating level chosen for each cost driver and the correspondent effort multiplier. In the last row there is the total product of all the effort multiplier that we will use in the final formula.

Cost Driver	**Rating Level**	**Effort Multiplier**
RELY	Nominal	1
DATA	Very High	1.28
CPLX	Nominal	1
RUSE	Nominal	1
DOCU	High	1.11
TIME	High	1.11
STOR	Nominal	1
PVOL	Low	0.87
ACAP	Very High	0.71
PCAP	Nominal	1

APEX	Very Low	1.22
PLEX	Low	1.19
LTEX	High	1.09
PCON	High	0.90
TOOL	High	0.90
SITE	Extra High	0.80
SCED	Very High	1
Total product		0.998945

3.8. Effort computation

At this point we can compute the effort expressed in Person-Month by applying the COCOMO formula mentioned at the beginning of the section 3.

$$PM = A * Size^{E} * \prod_{i=1}^{n} EM_i = 2.94 * 7.774^{1.091} * 0.998945 = 27.5158\, PM \simeq 28\, PM$$

3.9. Duration estimation

We assume to have 140 working hours in a month so:

$$28\, PM = 28 * 140\, PH = 3920\, PH$$

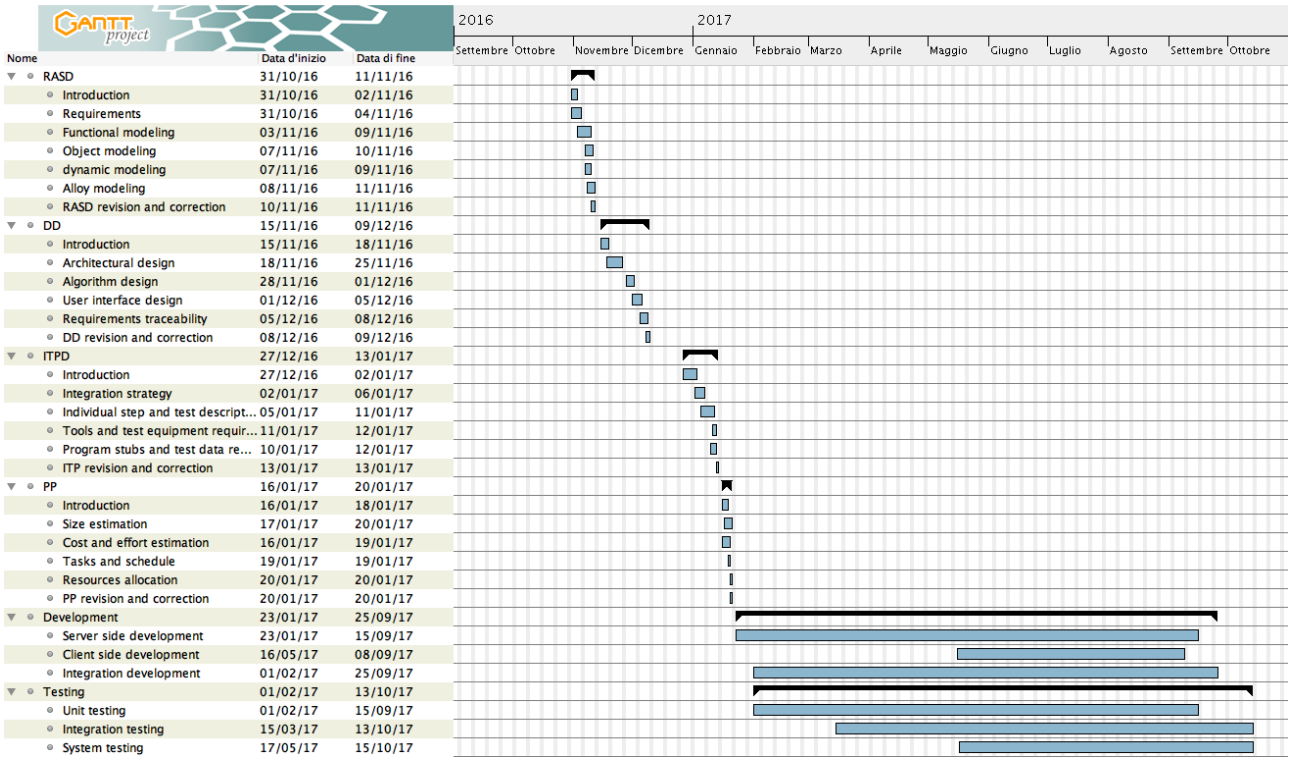
Since our team is composed of three people the total duration is:

$$\text{Duration} = \frac{3920\, PH}{3\, P} = 1306.66\, H$$

4. Tasks and schedule

We spent about 110 hours per person until now, so the work will be finished in approximately in 1195 hours, according to our estimation. Considering that, from now, we will start working 8 hours per-day:

$$\frac{1195}{140} \simeq 8{,}5\ Months$$

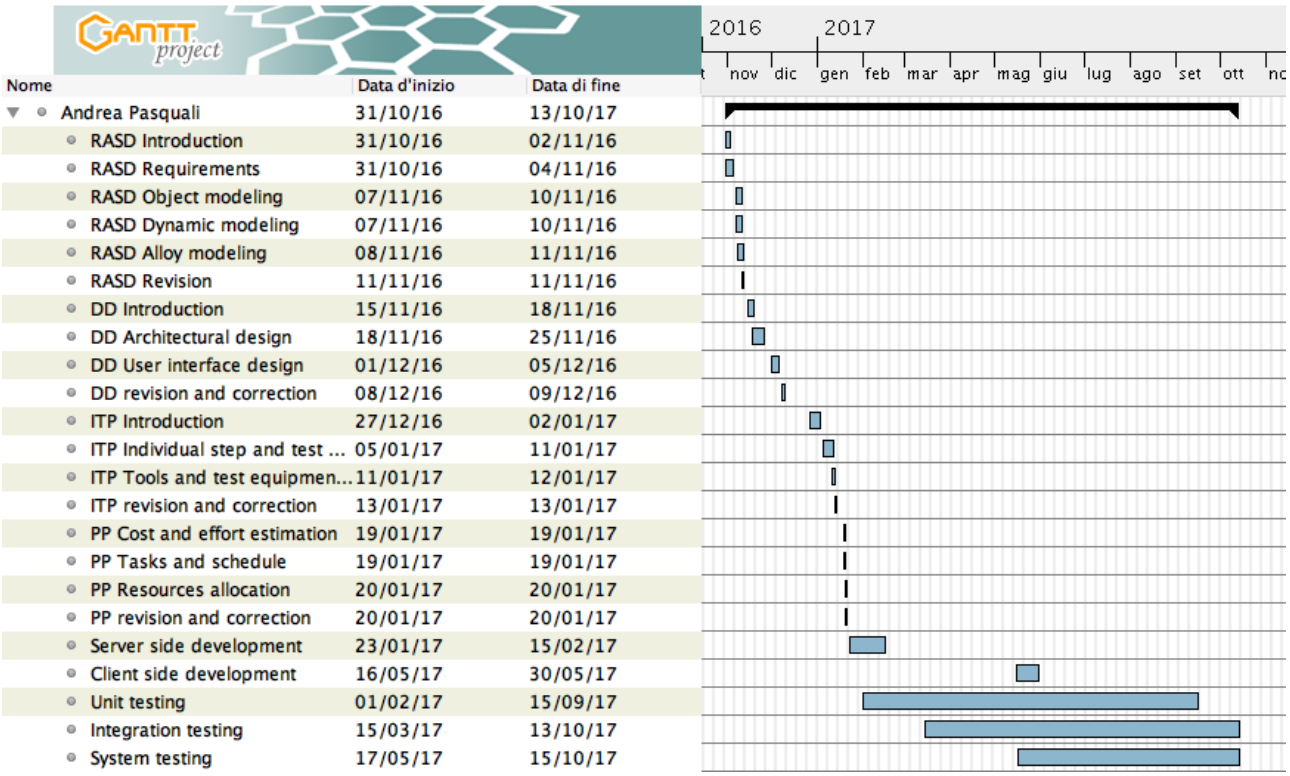


Nome	Data d'inizio	Data di fine
RASD	31/10/16	11/11/16
Introduction	31/10/16	02/11/16
Requirements	31/10/16	04/11/16
Functional modeling	03/11/16	09/11/16
Object modeling	07/11/16	10/11/16
dynamic modeling	07/11/16	09/11/16
Alloy modeling	08/11/16	11/11/16
RASD revision and correction	10/11/16	11/11/16
DD	15/11/16	09/12/16
Introduction	15/11/16	18/11/16
Architectural design	18/11/16	25/11/16
Algorithm design	28/11/16	01/12/16
User interface design	01/12/16	05/12/16
Requirements traceability	05/12/16	08/12/16
DD revision and correction	08/12/16	09/12/16
ITPD	27/12/16	13/01/17
Introduction	27/12/16	02/01/17
Integration strategy	02/01/17	06/01/17
Individual step and test descript...	05/01/17	11/01/17
Tools and test equipment requir...	11/01/17	12/01/17
Program stubs and test data re...	10/01/17	12/01/17
ITP revision and correction	13/01/17	13/01/17
PP	16/01/17	20/01/17
Introduction	16/01/17	18/01/17
Size estimation	17/01/17	20/01/17
Cost and effort estimation	16/01/17	19/01/17
Tasks and schedule	19/01/17	19/01/17
Resources allocation	20/01/17	20/01/17
PP revision and correction	20/01/17	20/01/17
Development	23/01/17	25/09/17
Server side development	23/01/17	15/09/17
Client side development	16/05/17	08/09/17
Integration development	01/02/17	25/09/17
Testing	01/02/17	13/10/17
Unit testing	01/02/17	15/09/17
Integration testing	15/03/17	13/10/17
System testing	17/05/17	15/10/17

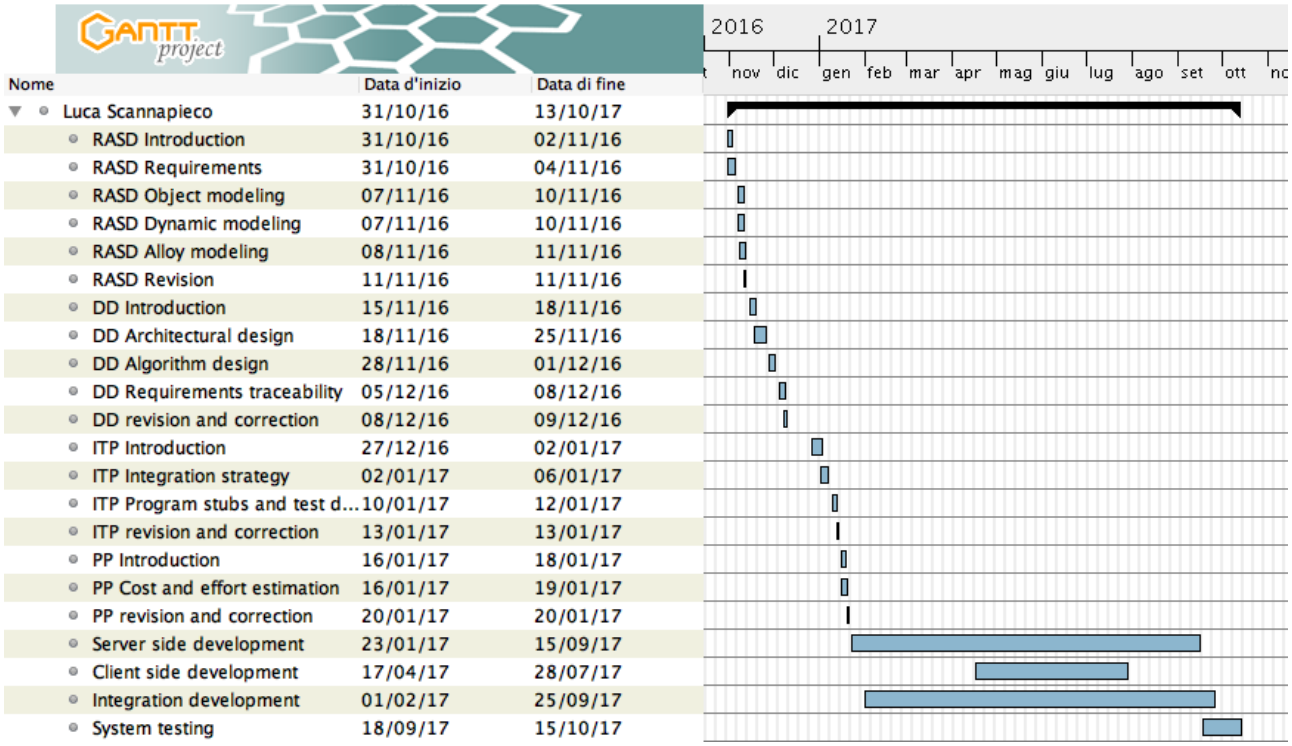
5. Resources allocation

Our team is composed of three people that often work together, so in resource allocation many activities are done from every component of the team.

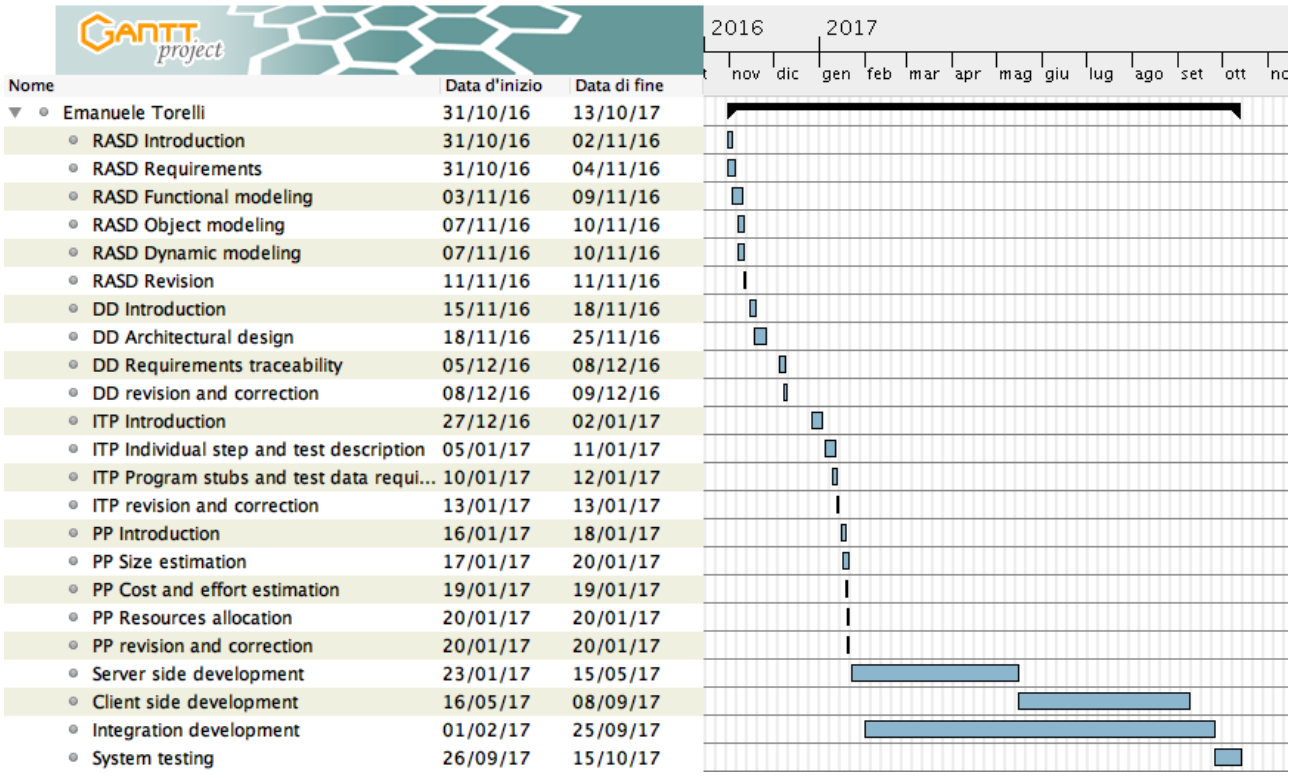
We divide the work of only some part of the project that are reported in the following tabs: parts that a member has not to work to aren't reported in the respective tab.



Nome	Data d'inizio	Data di fine
Andrea Pasquali	31/10/16	13/10/17
RASD Introduction	31/10/16	02/11/16
RASD Requirements	31/10/16	04/11/16
RASD Object modeling	07/11/16	10/11/16
RASD Dynamic modeling	07/11/16	10/11/16
RASD Alloy modeling	08/11/16	11/11/16
RASD Revision	11/11/16	11/11/16
DD Introduction	15/11/16	18/11/16
DD Architectural design	18/11/16	25/11/16
DD User interface design	01/12/16	05/12/16
DD revision and correction	08/12/16	09/12/16
ITP Introduction	27/12/16	02/01/17
ITP Individual step and test ...	05/01/17	11/01/17
ITP Tools and test equipmen...	11/01/17	12/01/17
ITP revision and correction	13/01/17	13/01/17
PP Cost and effort estimation	19/01/17	19/01/17
PP Tasks and schedule	19/01/17	19/01/17
PP Resources allocation	20/01/17	20/01/17
PP revision and correction	20/01/17	20/01/17
Server side development	23/01/17	15/02/17
Client side development	16/05/17	30/05/17
Unit testing	01/02/17	15/09/17
Integration testing	15/03/17	13/10/17
System testing	17/05/17	15/10/17



Nome	Data d'inizio	Data di fine
Luca Scannapieco	31/10/16	13/10/17
RASD Introduction	31/10/16	02/11/16
RASD Requirements	31/10/16	04/11/16
RASD Object modeling	07/11/16	10/11/16
RASD Dynamic modeling	07/11/16	10/11/16
RASD Alloy modeling	08/11/16	11/11/16
RASD Revision	11/11/16	11/11/16
DD Introduction	15/11/16	18/11/16
DD Architectural design	18/11/16	25/11/16
DD Algorithm design	28/11/16	01/12/16
DD Requirements traceability	05/12/16	08/12/16
DD revision and correction	08/12/16	09/12/16
ITP Introduction	27/12/16	02/01/17
ITP Integration strategy	02/01/17	06/01/17
ITP Program stubs and test d...	10/01/17	12/01/17
ITP revision and correction	13/01/17	13/01/17
PP Introduction	16/01/17	18/01/17
PP Cost and effort estimation	16/01/17	19/01/17
PP revision and correction	20/01/17	20/01/17
Server side development	23/01/17	15/09/17
Client side development	17/04/17	28/07/17
Integration development	01/02/17	25/09/17
System testing	18/09/17	15/10/17



Nome	Data d'inizio	Data di fine
Emanuele Torelli	31/10/16	13/10/17
RASD Introduction	31/10/16	02/11/16
RASD Requirements	31/10/16	04/11/16
RASD Functional modeling	03/11/16	09/11/16
RASD Object modeling	07/11/16	10/11/16
RASD Dynamic modeling	07/11/16	10/11/16
RASD Revision	11/11/16	11/11/16
DD Introduction	15/11/16	18/11/16
DD Architectural design	18/11/16	25/11/16
DD Requirements traceability	05/12/16	08/12/16
DD revision and correction	08/12/16	09/12/16
ITP Introduction	27/12/16	02/01/17
ITP Individual step and test description	05/01/17	11/01/17
ITP Program stubs and test data requi...	10/01/17	12/01/17
ITP revision and correction	13/01/17	13/01/17
PP Introduction	16/01/17	18/01/17
PP Size estimation	17/01/17	20/01/17
PP Cost and effort estimation	19/01/17	19/01/17
PP Resources allocation	20/01/17	20/01/17
PP revision and correction	20/01/17	20/01/17
Server side development	23/01/17	15/05/17
Client side development	16/05/17	08/09/17
Integration development	01/02/17	25/09/17
System testing	26/09/17	15/10/17

6. Risk management

In this section there is a list of all the risks that may affect the project and the possible troubles it may encounter.

First of all, we must be sure that with a car sharing service like PowerEnJoy, we are entrusting real cars to people. We can't ensure that PowerEnJoy users always drive responsively and accidents may happen, so the company has to necessarily stipulate an advantageous insurance policy to cope with steals, accidents, fire, or other damages to cars. Anyway, the company assesses that during a ride, the responsibility of the car is given only to the pending user. In any case the company will necessarily hire some operators to deal with accidents or cars left parked out of battery around the city.

Another important consideration is that users can be rude-behaving while using PowerEnJoy cars, therefore, in addition to a normal maintenance of the bodywork and internals of cars, also the devices that run the car app are subject to damages produced by a massive usage from the users (e.g. they can break the touchscreen while tapping on it too violently).

We also have to consider that PowerEnJoy isn't the only car sharing system available in Milano, so the society has to be ready to promote the service with consistent strategies of marketing. For the same reason, it must be considered that other companies can constantly improve their services by cutting costs and improve the functionalities of the system, so the society has to be ready to revise some of the policies in order to maintain faith of the clients. As for the PowerEnJoy system development, we believe that a friendly user interface and the facility of usage of the application can help in a very consistent way the spreading of the service among citizens, so we will try to achieve the most appetizing product possible.

A problem that can occur during the implementation of the system is that we must know how to design and develop software for different platforms that obviously require different programming languages and programming paradigms. In fact, only to develop the client-side apps, an iOS programmer with Swift knowledge for Apple devices is required, an Android programmer with Java knowledge is required too, and eventually a Windows programmer with C# knowledge, in addition to a web programmer to develop the web app which requires knowledge of many languages like HTML, CSS, JavaScript and PHP. Since we don't have such different backgrounds in programming experience, a possible way to reduce this issue is to adopt a cross-platform strategy, like Cordova or Xamarin, with the benefit that the application of every mobile platform is written once in the same language and run everywhere.

Another issue is related to our dependency on external services and components. PowerEnJoy doesn't depend on many external services, but for example the mapping service is one of core functionalities of every client app. A change in the terms and conditions of the mapping service, or even just a modification of the APIs itself, could pose serious technical problems. A possible countermeasure is to choose the most reliable and open to third party software mapping service and design the code to be as portable as possible and with a great modularity and independence between components.

Of course, since the system relies on a huge quantity of written code, losing some parts of the source code would be a disaster, but in this case the issue is more easy to tackle, since the staff can rely on versioning control systems and backup services to prevent the loss of any data.

Another possible issue is to integrate the physical sensors and equipment of the cars with the PowerEnJoy system, the staff must be sure that before starting developing software with a certain programming language instead of others, the hardware must be integrated with a full compatibility. Probably the most critical component is the on-board device which runs the car app: the engineers have to choose the most reliable component and the easiest to integrate, obviously before starting developing the car app.

Another important thing is letting the stakeholders have an active role in the development of the project, in the requirement analysis and design phases as well as in the implementation phase. Activities in this direction may include periodical reviews and meetings, demonstrations, discussions on the interface design and so on. We have to be conscious that putting together the requirements and desires of the stakeholders may not be an easy task and that some negotiations are certainly going to be there.

A final problem may also emerge from issues with the project scheduling. Even though an initial overall schedule is provided in this document, it can't obviously take into account all the possible issues that may arise down the road. For this reason, some extra time has been allocated at the expected end of each major activity to allow for adjustments.

7. Other info

7.1. Sample documents

- Assignments AA 2016-2017.pdf
- COCOMO II Model Definition Manual on http://sunset.usc.edu/
- Function Point Training Booklet New on http://www.softwaremetrics.com/
- Documents previously provided:
    - PowerEnJoy – RASD.pdf
    - PowerEnJoy – DD.pdf
    - PowerEnJoy – ITPD.pdf
- Sample documents:
    - Project planning example document.pdf
- Course slides:
    - Project Management Basics + Advanced.pdf

7.2. Used tools

- Microsoft Word 2016, for the drafting of the ITPD
- Microsoft OneDrive, to allow concurrent editing
- GitHub, to store the project in a repo
- GanttProject, for the drawing of Gantt diagrams in section 5. and 6.

7.3. Hours of work

For redacting and writing the Project Plan Document we spent approximately 30 hours per person.

7.4. Changelog

No changes in the document for the moment.