Project plan document

POLITECNICO
MILANO 1863

Figure 1: Logo Politecnico di Milano

PowerEnjoy

Figure 2: Logo PowerEnjoy

- Maria Chiara Zaccardi
- Nicola Sosio
- Riccardo Redaelli

01/22/17

Contents

1

Introduction

1.1 Purpose and scope

Project planning is an iterative process that starts when you create an initial project plan during the project startup phase. It's an approved document used to guide both prject execution an d project control. The primary uses of the project plan are to document planning assumptions and decisions, facilitate communication among project stakeholders and document approved scope, cost, and schedule baselines. A Project Plan contains an estimation of the costs and effort required to reach the project goals, we are going to use the function point and COCOMO for estimations in terms of lines of code and real cost required to develop it. Then we are going to divide the project into tasks and allocate each task to a timescale and to a member of the group. Finally we are going to evaluate possible risks that the project development could run into.

1.2 Definitions

- **Task**: activities which must be completed to achieve the project goal
- **Function point**: function point is a "unit of measurement" to express the software dimension

1.3 Abbreviations

- **RASD**: Requirement analysis and specification document
- **DD**: Design Document
- **ITPD**: Integration testing plan document
- **PP**: Project Plan Document

- **FSM**: functional size measurement
- **FP**: function point UFP: Unadjusted function point
- **SLOC**: source lines of code
- **4GL**: fourth generation programming language
- **AVC**: language-dependent factor varying from 200-300 for assembly language to 2 - 40 for a 4GL
- **ILF**: Internal Logic Files
- **EIF**: External Interfaces Files
- **EI**: External Input
- **EO**: External Output
- **EQ**: External Inquiry
- **GPS**: Global Positioning System
- **RELY:** Required Software Reliability
- **DATA:** Database Cost Driver
- **CPLEX:** Product Complexity
- **RUSE:** Required reusability
- **DOCU:** Documentation match life-cycle needs
- **TIME:** Execution time constraints
- **STOR:** Storage constraints
- **PVOL:** Platform volatility
- **ACAP:** analyst capability
- **PCAP:** Pragrammer capability
- **PCON:** Personnel Continuity
- **APEX:** Application experience
- **PLEX:** Platform experience
- **LTEX:** Language and tool experience
- **TOOL:** Usage of software tools
- **SITE:** Multisite development collocation
- **SCED:** Required development schedule

1.4 Reference Documents

- **RASD** produced before v1.2
- **DD** produced before v 1.1
- **ITDP** produced before
- Specification Document: Assignments AA 2016-2017.pdf
- Example from last year: Project planning example document.pdf
- Slides of lecture lessons from Beep: Project Management Basics+Advanced.pdf

2

Project size, cost and effort estimation

2.1 Size estimation: function points

A function point is a "unit of measurement" to express the software dimension. The dimension of software can be characterized based on the functionalities that it has to offer. Function point are used to compute a FSM of a software.

It's based on a combination of program characteristics such as data structures, inputs, outputs, inquiries and external interfaces.

The weight is associated with each of these FP counts and his estimation is calculated through statistical analysis from real projects.

Multiplying each "raw" count by the weight and summing all partial values we obtain the total.

$$UFP = \sum (N_i * w_i) \tag{2.1}$$

where N_i is the number of element of a given type and $w_i$is the associated weight.

Complexity is evaluated based on the characteristics of the application.

Function Type	Function Complexity		
	Simple	Medium	Complex
External Input (EI)	3	4	6
External Output (EO)	4	5	7
External Inquiry (EQ)	3	4	6
Internal Logical File (ILF)	7	10	15
External Interface Files (EIF)	5	7	10

Table 2.1: Function complexity of each function type

Function points can be used to calculate SLOC depending on the average number of SLOC per FP for a given language.

$$SLOC = AVC * FP \tag{2.2}$$

2.1.1 External Inputs

Input operations are elementary operation to elaborate data coming from the external environment. PowerEnJoy supports many kind of interactions with different users. In this section we are going to analyze the main functionalities and their corresponding complexity.

Concerning the **visitor** user:

- **[EI1] Registration**: This is an operation that needs to check a quite large number of parameters that the user has to insert. User has to insert Name, Surname, Email address, PhoneNumber, Username, Address, Number of Credit Card and Number of Driving Licence. The system has to check the validity of the inserted fields, send the back to the user the password (riferimento all'EO) for the login and register all data. This operation is evaluated as medium difficulty, therefore the FPs are 4.

Concerning the **user**:

- **[EI2] Login**: This operation needs to check just two parameters that the user has to insert: email and password. This is a simple operation so its FPs are 3.
- **[EI3] Make a reservation**: This operation imply the creation of the object in the database, associated to the correspondent user and car, then the system must change the status of the car from available to unavailable and start the timer that must last for at most one hour. It produces 4 FPs.
- **[EI4] Cancel reservation**: The operations involved in this action are the same that are involved in the reservation operation, unless for the timer. It produces 4 FPs.
- **[EI5] Enable money saving option**: Once the system has received the final destination address from the user, the system has to calculate the best power grid station where the user can leave the car in order to receive a discount. The distribution of the cars in the city is also a contributing factor in the calculation. Therefore this is a complex operation that requires 6 FPs.
- **[EI6] Report damage**: This operations only imply the car status change to damage car, but involved a quite large number of component for retrieve informations about the last ride and the last user. Then the system take care to advise the nearest operator through a push notification. This is a quite complex operation and its FPs are 6.

- **[EI7] Plug the car**: When a user plug the gar in a power grid station the system has to change the status of the car to unavailable until the car reach the 100% of battery. This is a simple operation and has a contribute of 3 FPs.
- **[EI8] Unlock the car (insert the code)**: This is a simple operation that involves only some component for check if the reservation is valid and the code that the user has to insert is the right one. The the CarSystem component unlock the car. It produces 3 FPs.

Concering the **operator**:

- **[EI9] Login**: This operation needs to check just two parameters that the user has to insert: username and password. This is a simple operation so its FPs are 3.
- **[EI10] Accept an operation**: This is a simple operation that let the operator view more informations about the car that he has to repair or move. It produces 3 FPs.
- **[EI11] Report operation informations**: This is also a simple operation that let the operator modify the information about the operation in order to add a description of what he had done and the total cost of the operation. It produces others 3 FPs.

External Input	Complexity	FPs
Registration	Simple	3
Login	Simple	3
Make a reservation	Medium	4
Cancel a reservation	Medium	4
Enable money saving option	Complex	6
Report damage	Complex	6
Plug the car	Simple	3
Unlcok the car (insert the code)	Simple	3
Login	Simple	3
Accept an operation	Simple	3
Report operations informations	Simple	3
Total		41

Table 2.2: Function complexity of external inputs

2.1.2 External Outputs

External outputs are elementary operation that generates data for the external environment. As a part of its normal behaviour, PowerEnJoy occasionally needs to communicate with the user outside the context of an inquiry.

- **[EO1] Notify an operator** that there is an operation to do. This is a simple operation so it produces just 4 FPs.
- **[EO2] Provide a password** for the login to the user when he/she has completed the registration phase. This is also a simple operation that contributes 4 FPs.
- **[EO3] Apply possible fee or discount** to the user at the end of the ride. This is a quite complex operation because the system has to check an elevate number of parameters in order to verify if the user can receive a discount or fee. It requires 7 FPs.
- **[EO4]** Show **during the ride** the map of the city with related power grid station and the outline of the safearea and some informations about the ride such as the duration. We consider this operation quite complex for the numerous data that the system has to retrieve. It's contribution is of 7 FPs.
- **[EO5]** Show **the total amount** at the end of the ride. This is a easy operation in fact the system has already calculate the possible fee or discount. This requires 4 FPs.

External Output	Complexity	FPs
Notify an operator	Simple	4
Provide a password	Simple	4
Apply possible fee or discount	Medium	7
Show the map	Medium	7
Show the total amount	Simple	4
Total		26

Table 2.3: Function complexity of external outputs

2.1.3 External Inquiries

External Inquiries are elementary operations that not require a significant elaboration of internal logic files. An inquiry is essentially a data retrieval request. Concerning **user**:

- **[EQ1] Select a car**: When a user select a car from the available cars, the system shows the car informations as the battery level and the possibility of reservation. This is a very simple operation and it contributes 3 FPs.
- **[EQ2] See Reservation informations**: Once the user has reserved a car, the system shows the information about user's reservation with the position of the car, the time left and the possibility of cancel reservation, unlock the car and report a damaged. This is a simple operation that requires 3 FPs.
- **[EQ3] See available car (by GPS)**: User search for the available cars activating the GPS. Then the system shows the available cars around his position. This is a simple operation and it contributes 3 FPs.
- **[EQ4] See available car (by address)**: This operation contributes the same FPs as the previous one. In fact this action involved the same component as the operation mentioned before, but instead of enable GPS the user has to insert the address.
- **[EQ5] See Power Grid Station and their position**: When user choose to see the power grid stations, the system shows him all power grid stations and their positions in the map of the city. This operation requires 3 FPs.
- **[EQ6] Availability of each Power Grid Station in real time**: For each power grid station the user can see the availability in order to know if he/she can leave the car at that power grid station. It contributes 3 FPs.
- **[EQ7] See Safe Area**: When user choose to see the safe area, the system shows him/her the outer limit. 3 FPs.
- **[EQ8] Amount of the cost of the ride**: During the ride the system shows to the user the amount of the total cost of the ride. This is simple operation that needs 3 FPs.

Concerning **operator**:

- **[EQ9] See unavailable cars that need an operation**: As users can see the available cars, operators can see the unavailable ones. It's the same operation instead of the fact that the system shows the unavailable cars. It's contribute is again 3 FPs.

External Inquiry	Complexity	FPs
Select a car	Simple	3
See reservation informations	Simple	3
See available car (by GPS)	Simple	3
See available car (by address)	Simple	3
See power grid station and their position	Simple	3
Availability of each Power Grid Station in real time	Simple	3
See Safe Area	Simple	3
Amount of the cost of the ride	Simple	3
See unavailable cars that need an operation	Simple	3
Total		27

Table 2.4: Function complexity of external inquiries

2.1.4 Internal Logic Files

Internal logic files are data used and managed by the application.

- **[ILF1]** The system has to manage the **information about the registered users**: Name, Surname, Email address, PhoneNumber, Username, Address, Number of Credit Card, Number of Driving Licence and the password that the system provide at the end of the registration phase. This is just a store operation, so it contributes 7 FPs.
- **[ILF2]** The system has to manage the **information about the registered operators**: Name, Surname, IDNumber and Password. IdNumber and Password are provided from the PowerEnJoy during the hired phase. Operators have also the position always available. This is just a store operation, so it contributes again 7 FPs.
- **[ILF3] SafeArea** is a set of positions as pairs of latitude and longitude points. So it contains a very large number of elements but they are all the same record type, so its complexity is medium and contributes 10 FPs.
- **[ILF4] PowerGrid Station** is a record type that contains the position and the availability. it's a simple record and produces 7 FPs.
- **[ILF5] Map** entity contains a list of the availability cars, a list of power grid stations and the Safe Area. This is a record that contains a large number of records, so it's a high complexity and contributes 15 FPs.
- **[ILF6]** The system has to manage **car informations** which contains a String with the plate of the car, IdCode, battery level, a boolean with the availability, a boolean for the damage, the position and number of

sensor that are active in order to measure the number of passengers. The complexity is low and the FPs produced are 7.

- **[ILF7] Reservation** needs to store the user that take the reservation, the car associated, time in which the user reserve the car and the timer associated. The complexity is low and produces 7 FPs.
- **[ILF8] Ride** needs to store the driver user, the car associated, number of passengers, the total time, starting time and a boolean that is true when the ride is ended. The complexity is again low, and requires 7 FPs.
- **[ILF9]** The system has to manage also the **operation** entity, which include informations about the operator, the charged user and the damage car. The system can register three type of operations: Move operation, Re-charge operation and Restoration operation. This is a medium complexity operation and requires 10 FPs.

Internal Logic File	Complexity	FPs
Information registerd user	Simple	7
Information registered operator	Simple	7
Safe area	Complex	10
Power grid stations	Simple	7
Map	Complex	15
Car informations	Simple	7
Reservation	Simple	7
Ride	Simple	7
Operations	Medium	10
Total		77

Table 2.5: Function complexity of internal logic files

2.1.5 External Logic Files

External interface files are data generated and maintained by other applications but they are sent to our application in order to use them. Since those operations need to interface with external services and to retrieve a large amount of data, they are considered as complex operations.

- **[EIF1] GPS signal**: the system needs to have access to the GPS signal in order to find the location of each car and each operator.
- **[EIF2] Reverse geocoding**: When the user input an address the system needs to get the correspondent pair of coordinates.
- **[EIF3] Map data retrieval**: the mapping service is used to retrieve the graphical representation of the city map to be displayed on the onboard screen on the car.

External Logic File	Complexity	FPs
GPS signal	Complex	10
Reverse geocoding	Complex	10
Map data retrieval	Complex	10
Total		30

Table 2.6: Function complexity of internal logic files

2.1.6 Overall estimation

Function Type	Function Complexity
External Input (EI)	41
External Output (EO)	26
External Inquiry (EQ)	27
Internal Logical File (ILF)	77
External Interface Files (EIF)	30
Total	201

Table 2.7: Function complexity of each function type

Using Java EE as a development platform the total amount of SLOC are estimated through the formula 2.2 :

$$SLOC = 46 * 201 = 9246 \tag{2.3}$$

2.2 Cost and effort estimation : COCOMO II

In this section we're going to use the COCOMO II approach to estimate the cost and effort needed to develop PowerEnjoy.

Eary Design is aimed at an early estimation, since our project is not an early phase of the we will use a Post-Architecture cost model.

The COCOMO II effort estimation model is used to estimate the effort expressed in person month. The number of person-month in COCOMO II model is computed as follow:

$$PM = A * Size^{E} * \prod_{i=1}^{n} EM_i \tag{2.4}$$

Where:

- $A = 2.94$ approximates a productivity constant in ${}^{PM}/_{KSLOC}$.
- $Size$ has been estimated in the Section 2.1.6.
- EM_i Effort Multiplier.

$$E = B + 0.01 * \sum_{j=1}^{5} SF_j \tag{2.5}$$

Where:

- $B = 0.91$
- SF_j Scale Factor

2.2.1 Scale drivers

In order to compute the exponent E (2.5) we have to choose for each scale factor the rating level by referring to the following COCOMO table.

Each choice has been done after taking into consideration several aspects for each scale factor explained in the following sections.

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
Prec	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar 2.48	largely familiar 1.24	throughly familiar
SF_j	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_j	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_j	7.07	5.65	4.24	2.83	1.41	0.00
Team	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
SF_j	5.48	4.38	3.29	2.19	1.10	0.00
Pmat	Level Lower 1	Level Upper 1	Level 2	Level 3	Level 4	Level 5
SF_j	7.80	6.24	4.68	3.12	1.56	0.00

Table 2.8: Scale Factors for COCOMO II Post-Architecture Models

2.2.1.1 Precedentedness

Precedentedness reflects the similarity of the project with previous large scale projects. If a product is similar to several previously developed projects, then the precedentedness is high.

Feature	Very Low	Nominal/High	Extra High
Organizational understaiding of pruduct objectives	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

Table 2.9: Precedentdedness scale factor for COCOMO II

2.2.1.2 Development flexibility

Development flexibility reflects the degree of flexibility in the development of the process with respect to constraints to conform requirements and user interface.

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
Premium on early completion	High	Medium	Low

Table 2.10: Development flexibility scale factor for COCOMO II

2.2.1.3 Risk resolution

Feature	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
Percent of required top software architects available to project	20	40	60	80	100	120
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

Table 2.11: Risk resolution scale factor for COCOMO II

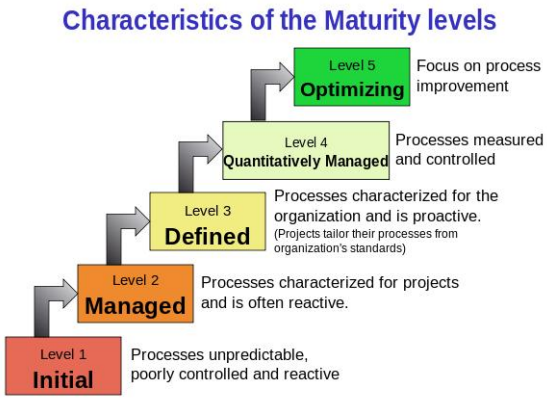
2.2.1.4 Team cohesion

Team cohesion scale factor accounts for difficulties in synchronizing the project's stakeholders: users, customers, developers, maintainers, interfacers, others.

Feature	Very Low	Low	Nominal	High	Very High	Extra High
Consistency of stakeholder objectives and cultures	Little	Some	Considerable	Generally	Strong	Full
Ability, willingness of stakeholders to accommodate other stakeholders' objectives	Little	Some	Considerable	Generally	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

Table 2.12: Team cohesion scale factor for COCOMO II

2.2.1.5 Process maturity



Rating	Maturity Level
Very Low	CMM Level 1 (lower half)
Low	CMM Level 1 (upper half)
Nominal	CMM Level 2
High	CMM Level 3
Very high	CMM Level 4
Extra high	CMM Level 5

Table 2.13: Process maturity scale factor for COCOMO II

2.2.1.6 Overall risult

The result of the analysis of the scale drivers is the following:

$$\sum_{j=1}^{5} SF_j = 17.85 \tag{2.6}$$

Feature	Factor	Value
Precedentedness	Very Low	6.20
Development flexibility	Nominal	3.04
Risk resolution	High	2.83
Team cohesion	Very High	1.10
Process maturity	Nominal	4.68
Total		17.85

Table 2.14: Scale factor result

2.2.2 Cost Drivers

2.2.2.1 Required software reliability

This is the measure of the effects of a failure in the system. If the effect of a software failure is only slight inconvenience then RELY is very low. If a failure would risk human life then RELY is very high. Since a failure in our system cause no high losses RELY cost driver is set to Low.

Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
Rely Descriptors	slightly inconvinience	easily recoverable losses	moderate recoverable losses	high financial losses	risk to human life	
Effort Multipliers	0.82	0.92	1.00	1.10	1.26	n/a

Table 2.15: RELY Cost Drivers

2.2.2.2 Database size

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. The reason the size of the database is important to consider is because of the effort required to generate the test data that will be used to exercise the program. Since our project does not foresee a large ammount of data test DATA rating level is setted to Nominal.

Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
Data Descriptors		$\frac{D}{P} < 10$	$10 < \frac{D}{P}$ $\frac{D}{P} < 100$	$100 < \frac{D}{P}$ $\frac{D}{P} < 1000$	$\frac{D}{P} > 1000$	
Effort Multipliers	n/a	0.9	1.00	1.14	1.28	n/a

Table 2.16: DATA Cost Drivers

2.2.2.3 Product complexity

This is the measure of the software complecity, CPLX cost driver rating is based on five different areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations. After analyzing each specific area we reach the conclusion that the CPLX rating for our system should be setted to Nominal.

Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.73	0.87	1.00	1.17	1.34	1.74

Table 2.17: CPLEX Cost Drivers

	Control Operations	Computational Operations	Device-dependent Operations	Data Management Operations	User Interface Operations
Very Low	Straight-line code with a few non-nested structured programming operators: DOs, CASEs, IF-THEN-ELSEs.	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.
Nomina	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.

Table 2.18: CPLX Ratings Levels

	Control Operations	Computational Operations	Device-dependent Operations	Data Management Operations	User Interface Operations
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	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, natural language interface.

Table 2.19: CPLX Ratings Levels

2.2.2.4 Required reusability

This cost driver takes into consideration the additional effort needed to construct components intended for reuse on current or future projects. 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. Since our components are supposed to be reusable within the project RUSE rating level is setted to Nominal.

Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
Ruse Descriptors		None	Across project	Across program	Across product line	Across multiple product lines
Effort Multipliers	n/a	0.95	1.00	1.07	1.15	1.24

Table 2.20: RUSE Cost Drivers

2.2.2.5 Documentation match to life-scycle needs

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). Since we have produced the properly documentation almost for all the phases of the product life-cycle DOCU rating level is setted to Nominal.

Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
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	
Effort Multipliers	0.81	0.91	1.00	1.11	1.23	n/a

Table 2.21: DOCU Cost Drivers

2.2.2.6 Execution time constraint

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 or subsystem consuming the execution

time resource. Since our system has to manage multiple requests TIME rating level is setted to High.

Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
Time Descriptors			≤ 50% use of available execution time	70%	85%	95%
Effort Multipliers	n/a	n/a	1.00	1.11	1.29	1.63

Table 2.22: TIME Cost Drivers

2.2.2.7 Storage constraint

This rating represents the degree of main storage constraint imposed on a software system or subsystem. Since nowadays the storage constraint is not still relevant STOR rating level is setted to Nominal.

Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
Stor Descriptors			≤ 50% use of available storage	70%	85%	95%
Effort Multipliers	n/a	n/a	1.00	1.05	1.17	1.46

Table 2.23: STOR Cost Drivers

2.2.2.8 Platform volatility

"Platform" is used here to mean the complex of hardware and software (OS, DBMS, etc.) the software product calls on to perform its tasks. 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. Since our software volatility depends on the volatility of the mobile operating system and mobile operating system will be usually updated onece a year PVOL is setted to Low.

Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
Rely Descriptors		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	
Pvol Multipliers	n/a	0.87	1.00	1.15	1.30	n/a

Table 2.24: PVOL Cost Drivers

2.2.2.9 Analyst capability

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. Since the requirement analysis and the design anaylsis have been propery done by analysts ACAP rating level is setted to High.

Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
Acap Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Effort Multipliers	1.42	1.19	1.00	0.85	0.71	n/a

Table 2.25: ACAP Cost Drivers

2.2.2.10 Programmer capability

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. Since this is the first project PCAP rating level could not be high but we have already worked as a group in a previous software project the ability to cooperate is high. The PCAP rating level is setted to Nominal.

Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
Pcap Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Effort Multipliers	1.34	1.15	1.00	0.88	0.76	n/a

Table 2.26: PCAP Cost Drivers

2.2.2.11 Personnel continuity

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. Since there is no personal turnover in the project the PCON rating level is settede toVery High.

Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
Pcon Descriptors	48% / year	24% / year	12% / year	6% / year	3% / year	
Effort Multipliers	1.29	1.12	1.00	0.90	0.81	n/a

Table 2.27: PCON Cost Drivers

2.2.2.12 Application experience

The rating for APEX is dependent on the level of applications experience of the project team developing the software system or subsystem. Since our team have no experience in development of Java EE system APEX rating level is setted to Very Low.

Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
Apex Descriptors	$\leq$ 2 months	6 months	1 year	3 years	6 years	
Effort Multipliers	1.22	1.10	1.00	0.88	0.81	n/a

Table 2.28: APEX Cost Drivers

2.2.2.13 Platform experience

The rating for PLEX account the importance of understanding the use of more powerful platforms, including more graphic user interface, database, networking, and distributed middleware capabilities. Since our team have no experience in Java EE platform PLEX rating level is setted to Very Low.

Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
Plex Descriptors	≤ 2 months	6 months	1 year	3 years	6 years	
Effort Multipliers	1.19	1.09	1.00	0.91	0.85	n/a

Table 2.29: PLEX Cost Drivers

2.2.2.14 Language and tool experience

This is a measure of the level of programming language and software tool experience of the project team developing the software system or subsystem. Software development includes the use of tools that perform requirements and design representation and analysis, configuration management, document extraction, library management, program style and formatting, consistency checking, planning and control, etc.

Since our team have no experience in development of Java EE system but already have Java knowledge LTEX rating level is setted to Low.

Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
Ltex Descriptors	≤ 2 months	6 months	1 year	3 years	6 years	
Effort Multipliers	1.20	1.09	1.00	0.91	0.84	n/a

Table 2.30: LTEX Cost Drivers

2.2.2.15 Usage of software tools

This is a measure of the knowledge of software tools. The tool rating ranges from simple edit and code, very low, to integrated life-cycle management tools, very high. Since our team have already used IDE like Eclipse which helps developers during almost all the phases TOOL rating level is setted to High.

Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
Tool Descriptors	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	
Effort Multipliers	1.17	1.09	1.00	0.90	0.78	n/a

Table 2.31: TOOL Cost Drivers

2.2.2.16 Multisite development collocation

Determining its 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). Since our team is allocated within the same metro area and the comunication among team's member have been done through internet services SITE rating level is setted to High.

Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
Collocation Descriptors	International	Multi-city and multi-company	Multi-city or multi-company	Same city or metro area	Same building or complex	Fully collocated
Comunication Descriptors	Some phone, mail	Individual phone, FAX	Narrow-band email	Wide-band electronic communication.	Wide-band elect. comm, occasional video conf.	Interactive multimedia
Effort Multipliers	1.22	1.09	1.00	0.93	0.86	0.80

Table 2.32: SITE Cost Drivers

2.2.2.17 Required development schedule

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. Since our team spend a lot of effort into the first phase SCED rating level is setted to High.

Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
Sced Descriptors	75% of nominal	85% of nominal	100% of nominal	130% of nominal	160% of nomina	
Effort Multipliers	1.43	1.14	1.00	1.00	1.00	n/a

Table 2.33: SCED Cost Drivers

2.2.2.18 Overall risult

Feature	Factor	Value
Required Software Reliability	Low	0.92
Database size	Nominal	1.00
Product complexity	Nominal	1.00
Required Reusability	Nominal	1.00
Documentation match to life-cycle needs	Nominal	1.00
Execution Time Constraint	High	1.11
Main storage constrain	Hominal	1.00
Platform volatility	Low	0.87
Analyst capability	High	0.85
Programmer capability	Nominal	1.00
Personnel continuity	High	0.81
Application Experience	Very Low	1.22
Platform Experience	Very Low	1.19
Language and Tool Experience	Low	1.09
Usage of Software Tools	High	0.90
Multisite development	High	0.93
Required development schedule	High	1.00
Total		0.81

Table 2.34: Cost factor result

2.2.3 Effort quantification

Since all the scale drivers and all the costs have been estimated, the effort expressed in person-month can be computed through the formula2.5 and 2.4 :

$$E = B + 0.01 * \sum_{j=1}^{5} SF_j = 0.91 + 0.01 * 17.85 = 1.0885 \tag{2.7}$$

$$PM = A * Size^E * \prod_{i=1}^{n} EM_i = 2.94 * 9246^{1.0885} * 0.81 \backsimeq 27PM \tag{2.8}$$

2.2.4 Schedule estimation

Finally we can estimate the Time to Develop by using the following formula:

$$TDEV = [C * PM^F] * \frac{SCED\%}{100} = 3.67 * 27^{0.3157} * \frac{130}{100} \simeq 14months \tag{2.9}$$

$$F = D + 0.2 * (E - B) = 0.28 + 0.2 * (1.86 - 0.91) = 0.3157 \tag{2.10}$$

Where:

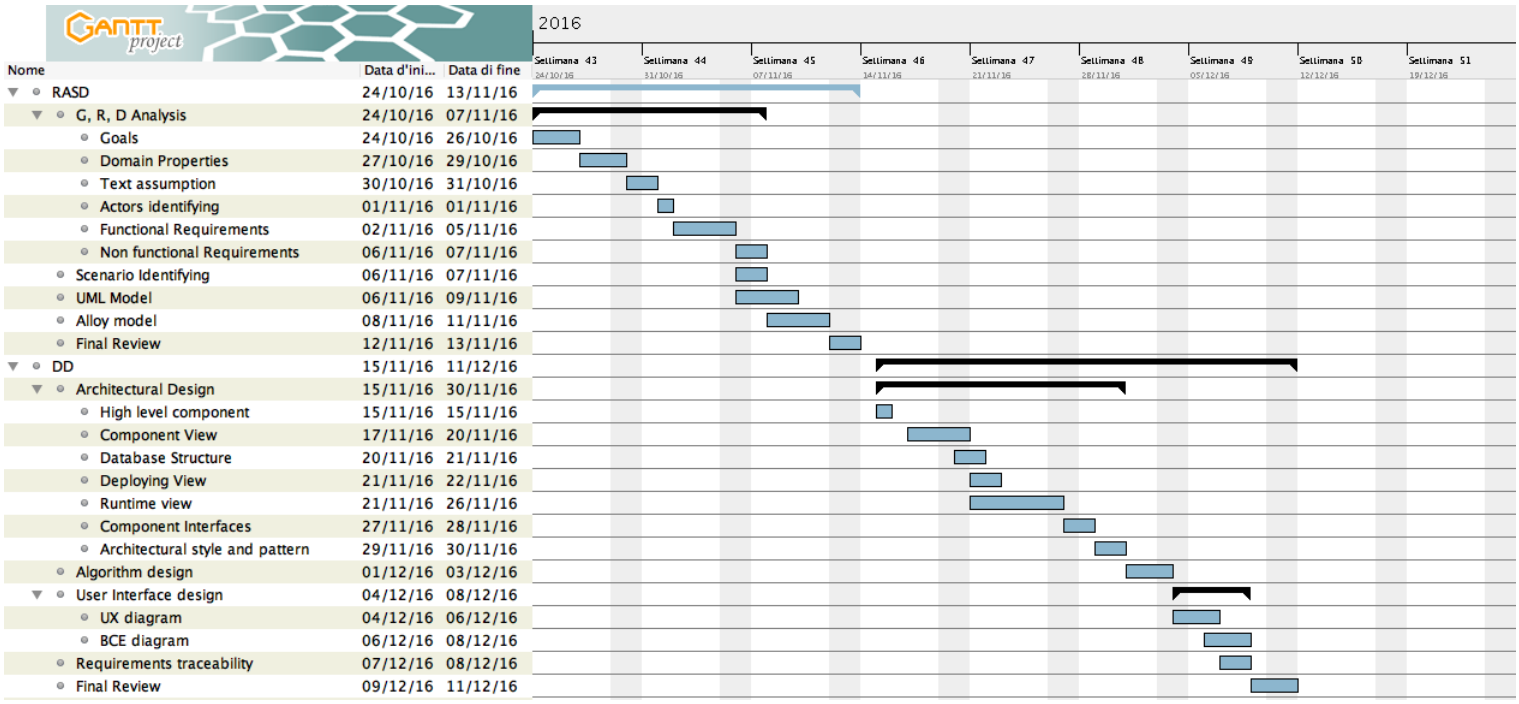
- $C = 3.67$.
- PM is the estimated person-month (2.8).
- $D = 0.28$.
- E is the effort scaling exponent (2.7) derived as the sum of project scale factors.
- $B = 0.91$.
- $SCED\%$ is the compression/expansion percentage in the SCED effort multiplier rating scale discussed in Section 2.2.2.17.

3

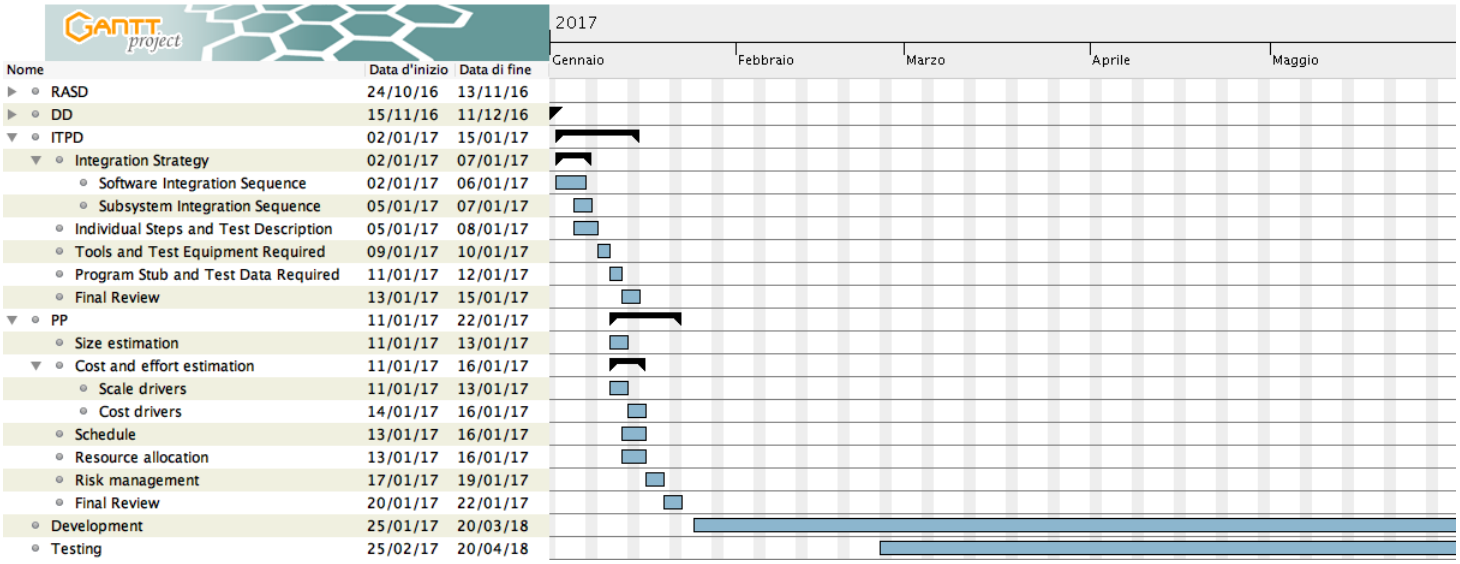
Schedule

In this section we are going to divide the project in different tasks. Each task correspond to a document: RASD, DD, ITDP and PP.

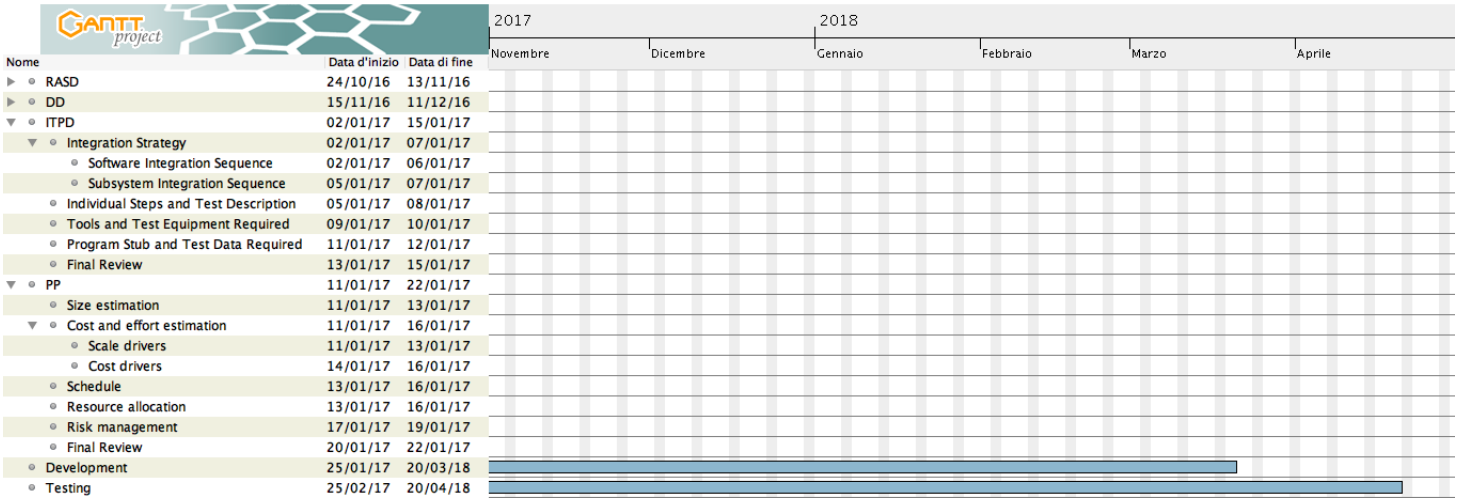
The last tasks is the development phase and testing phase. The development phase will last 14 months as estimated in the Section 2.2.4 (2.9).



Nome	Data d'ini...	Data di fine
RASD	24/10/16	13/11/16
G, R, D Analysis	24/10/16	07/11/16
Goals	24/10/16	26/10/16
Domain Properties	27/10/16	29/10/16
Text assumption	30/10/16	31/10/16
Actors identifying	01/11/16	01/11/16
Functional Requirements	02/11/16	05/11/16
Non functional Requirements	06/11/16	07/11/16
Scenario Identifying	06/11/16	07/11/16
UML Model	06/11/16	09/11/16
Alloy model	08/11/16	11/11/16
Final Review	12/11/16	13/11/16
DD	15/11/16	11/12/16
Architectural Design	15/11/16	30/11/16
High level component	15/11/16	15/11/16
Component View	17/11/16	20/11/16
Database Structure	20/11/16	21/11/16
Deploying View	21/11/16	22/11/16
Runtime view	21/11/16	26/11/16
Component Interfaces	27/11/16	28/11/16
Architectural style and pattern	29/11/16	30/11/16
Algorithm design	01/12/16	03/12/16
User Interface design	04/12/16	08/12/16
UX diagram	04/12/16	06/12/16
BCE diagram	06/12/16	08/12/16
Requirements traceability	07/12/16	08/12/16
Final Review	09/12/16	11/12/16



Nome	Data d'inizio	Data di fine
RASD	24/10/16	13/11/16
DD	15/11/16	11/12/16
ITPD	02/01/17	15/01/17
Integration Strategy	02/01/17	07/01/17
Software Integration Sequence	02/01/17	06/01/17
Subsystem Integration Sequence	05/01/17	07/01/17
Individual Steps and Test Description	05/01/17	08/01/17
Tools and Test Equipment Required	09/01/17	10/01/17
Program Stub and Test Data Required	11/01/17	12/01/17
Final Review	13/01/17	15/01/17
PP	11/01/17	22/01/17
Size estimation	11/01/17	13/01/17
Cost and effort estimation	11/01/17	16/01/17
Scale drivers	11/01/17	13/01/17
Cost drivers	14/01/17	16/01/17
Schedule	13/01/17	16/01/17
Resource allocation	13/01/17	16/01/17
Risk management	17/01/17	19/01/17
Final Review	20/01/17	22/01/17
Development	25/01/17	20/03/18
Testing	25/02/17	20/04/18

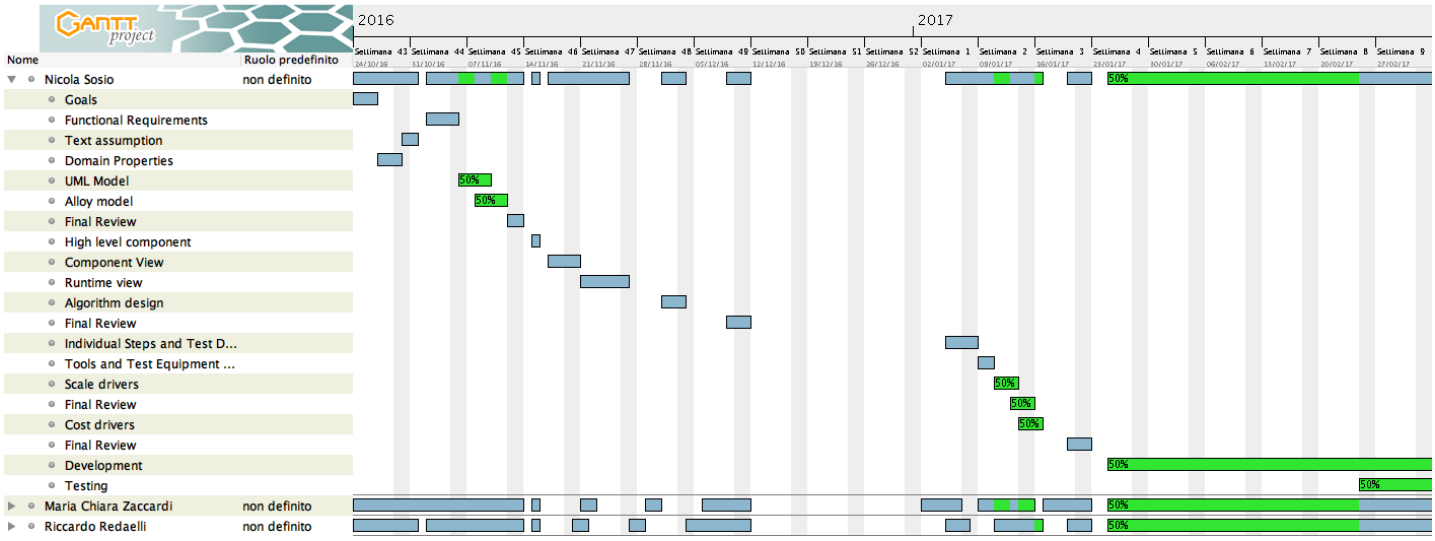


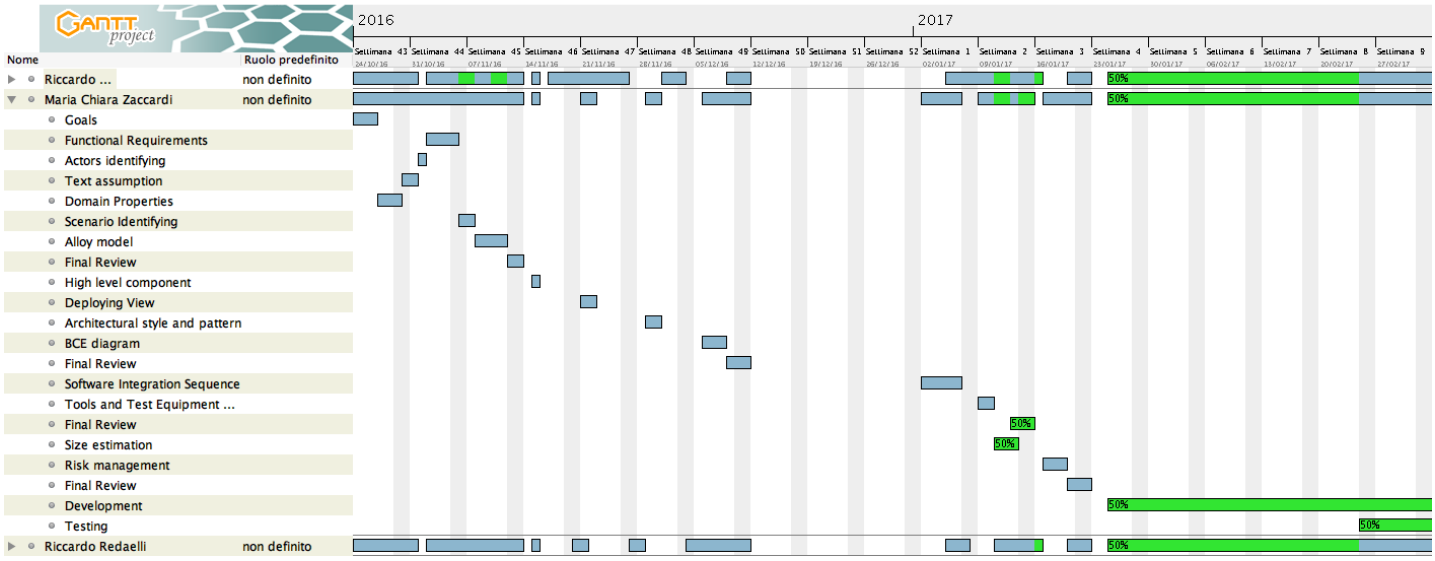
Nome	Data d'inizio	Data di fine
RASD	24/10/16	13/11/16
DD	15/11/16	11/12/16
ITPD	02/01/17	15/01/17
Integration Strategy	02/01/17	07/01/17
Software Integration Sequence	02/01/17	06/01/17
Subsystem Integration Sequence	05/01/17	07/01/17
Individual Steps and Test Description	05/01/17	08/01/17
Tools and Test Equipment Required	09/01/17	10/01/17
Program Stub and Test Data Required	11/01/17	12/01/17
Final Review	13/01/17	15/01/17
PP	11/01/17	22/01/17
Size estimation	11/01/17	13/01/17
Cost and effort estimation	11/01/17	16/01/17
Scale drivers	11/01/17	13/01/17
Cost drivers	14/01/17	16/01/17
Schedule	13/01/17	16/01/17
Resource allocation	13/01/17	16/01/17
Risk management	17/01/17	19/01/17
Final Review	20/01/17	22/01/17
Development	25/01/17	20/03/18
Testing	25/02/17	20/04/18

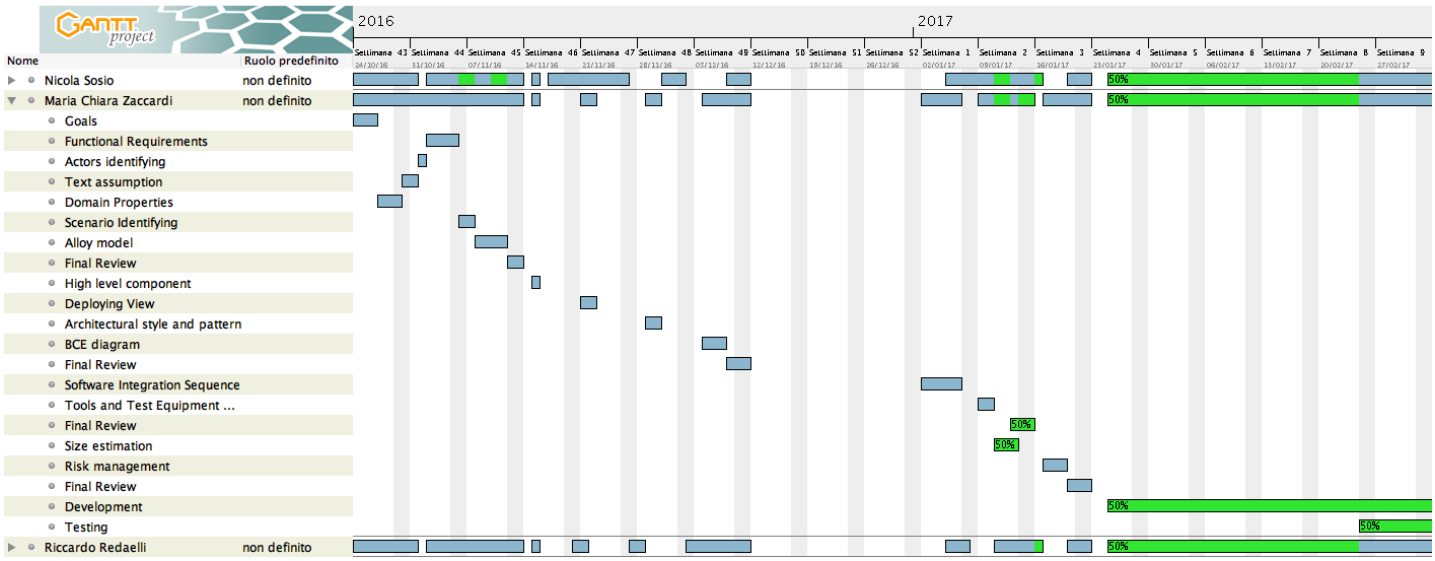
4

Resource allocation

In this section we are going to show how the tasks are divided. Our team is composed of three members and for the majority of the tasks that concerns important decision for the project we work together. We divide the work only for task that are independent each other.




GANTT project
2016
2017
Nome
Ruolo predefinito
Riccardo ... non definito
Maria Chiara Zaccardi non definito
Goals
Functional Requirements
Actors identifying
Text assumption
Domain Properties
Scenario Identifying
Alloy model
Final Review
High level component
Deploying View
Architectural style and pattern
BCE diagram
Final Review
Software Integration Sequence
Tools and Test Equipment ...
Final Review
Size estimation
Risk management
Final Review
Development
Testing
Riccardo Redaelli non definito
50%


GANTT project
2016
2017
Nome
Ruolo predefinito
Nicola Sosio non definito
Maria Chiara Zaccardi non definito
Goals
Functional Requirements
Actors identifying
Text assumption
Domain Properties
Scenario Identifying
Alloy model
Final Review
High level component
Deploying View
Architectural style and pattern
BCE diagram
Final Review
Software Integration Sequence
Tools and Test Equipment ...
Final Review
Size estimation
Risk management
Final Review
Development
Testing
Riccardo Redaelli non definito
50%

5

Risk management

In this section we have identified some potential risks that the project development may face, we identify three category of potential risks: technical, project and business issues.

A first threat belonging to project issues come from the possibility that committee change the requirements. This could be a marginal problem if changes are little, but if changes are important this may inevitably delay the overall project schedule. We have also consider that we could had done a bad schedule, so that tasks prevision may be unrealistic. This will cause an unavoidable delay in the project schedule. We have previously planned that each tasks must be always completed some day before milestone, so that possible slight problems arising in the course of the work may already have been budgeted.

Another issue related to this category is the management of data and the risk to corrupt or lost it. To prevent this risk PowerEnjoy have to use a version control system for tracking changes in computer files and coordinating work on those files among multiple people, such as using Git. Furthermore company should use also a backup services to prevent the loss of any data.

Another issue that must not be underestimated is related to our dependency on external services. A change in terms and conditions or a modification of their API could create to PowerEnjoy services serious technical problems, and this could cause also financial problems. One example of this API is the mapping service. A possible countermeasure is to design the code to be as portable as possible, so that it could have great modularity and independence between components.

The team must be sure that all sensor and equipment of the car are reliable and easy to integrate. The first problem we might have is the integration between sensors and programming languages, to avoid this we must be sure before start the project that they are compatible with our necessity, so that the hardware is integrated with a full compatibility.

Since the moment PowerEnjoy is a car sharing service we rent physical cars to people that sometimes have unpredictable behaviour. For this reason we need an advantageous policy to cope possible damage to things and people, for

example a fire and theft policy and also against act of vandalism. Anyway the company specified that during a ride the pending user take on the responsibility of driving.

We can identify other car sharing companies (other startup or companies already existing in the same city) as an important business risk to the success of PowerEnjoy because they reduce our business opportunities. In fact existing company could constantly enhance their services by cutting costs or improving or adding new functionalities on their system. So PowerEnjoy must be prepared to be competitive, adding or revising some policies, to make clients faithful customer. Probably a friendly user interface and the ease of usage of the application surely help the spreading of the service among citizens.

Another business risk can be identified in sponsoring the new service. It's not certain that once the project is completed there are sufficient user to have a profit. This problem is mainly due to the fact that in the city there are already other car sharing services.

PowerEnjoy require an aggressive advertising campaign, which may include discounts for new customers, advertising in the city and other conventions. So PowerEnjoy could entrust the management to another specialized company who could help it in promote and advertise the use of its service.

6

Used tools

- Github: for version controller
- Lyx: to format this document
- Google doc: for write the document
- GanttProject: for gantt diagrams

7

Effort Spent

For redacting and writing this document we spent 25 hours per person.