Politecnico di Milano

AA 2016-2017

Software Engineering 2 project: "*PowerEnJoy*"

Prof. Mottola Luca

PP

Project **P**lan

22/01/17 - version 1.0

Edited by:

Giulia Pennati **878686**

Paola Sanfilippo **809689**

Marco Siracusa **878083**

Table of Contents

1. Document structure 3
2. Function points 4
   - Internal Logical Files (ILF) 4
   - External Interface Files (EIF): 5
   - External Inputs: 5
   - External Outputs: 6
   - External Inquiries: 7
   - Overall Estimation 8
3. Cocomo 9
4. Task Identification 14
5. Schedule 16
   - 5.1. RASD and DD schedule 16
   - 5.2. Project schedule 17
6. Resource Allocation 18
7. Risk Management 19
8. Time effort 21

1. Document structure

We decided to divided the document in the following way:

- at first there's the definition and analysis of the function points, that are summed up in an overall estimation (in which the "Lines Of Code" parameter is computed).
- Then we used COCOMO to compute the effort and cost to carry out the software development process.
- After that, there's the identification of the tasks that are used for the schedule and the resource allocation.
- In the end we have the risk analysis, in which for each risk we provide

2. Function points

Note: *here a link to a good reference about function point (although not official) follows: http://www.functionpointmodeler.com/fpm-infocenter/index.jsp?topic=%2Fcom.functionpointmodeler.fpm.help%2Fditafiles%2Fconcepts%2Fcon-91.html*

Internal Logical Files (ILF)

- The **session** data structure is considered to be complex since it is supposed to store a great amount of records (estimated 25000 sessions/week/city) composed by a considerable amount of fields (13) of various complexity and size.
  Moreover, the data structure is in relation with four other data structures.
  However, the data content can be considered to be rather static since there are not so many read/write/update actions when the session is active and even less when the session is over (in average a session lasts between 20 and 60 minutes). It means that when the session is active, its data would be queried several times (not so often though) with a need of a quick response that, however, is unnecessary when the session is over.
- The **user**'s account data structure is considered to be medium complexity since a great amount of data (70000 records (users/city) each of 18 fields) is stored although those are rather statics; indeed, those are updated only when a user starts requires a reservation, approaches the car or rides that car.
  The data structure is in relation with another one.
- The **car**'s data structure is considered to be complex since it has to store the most up-to-date data (therefore frequently updated) of each car (supposed to be about 600 the carrying out 600 records with 10 fields each). Being the car shared between the users, those are supposed to be often running and then producing a great amount of new information of quite different nature to get stored and managed.
  The data structure is in relation with two others.
- The **power grid station** data structure contains a small amount of data (about 100 records (PGS/city) with 4 fields each) that are mainly of a static type despite a field updated slightly often (when a car gets plugged or unplugged).
  The data structure is in relation with two others.
- The **transaction** data structure is used to store (not so often) a small amount of data just at the end of a reservation or ride that are useful to track the payment history of the user (therefore there is no need of a quick response on the queries).
  The data structure is in relation with one other.

Logic File (ILF)	**Complexity**	**Weight**
Session	High	15
User	Medium	10
Car	High	15
Power Grid Station	Medium	10
Transaction	Low	07

Statistic data source: *http://www.milanotoday.it/green/mobilita/smart-car2go-statistiche-noleggi.html*

External Interface Files (EIF):

- No external files have been used here

External Inputs:

- The **registration request** is a quite complex operation that is supposed to involve several components and interactions (as it has also to create a new account, as instance).
- The **login procedure** is a pretty easy operation that involves just the database and not so many other components.
- The **logout procedure** is worth as above because of the same reasons.
- The **start ride procedure** is considered to be an easy operation since it has just to communicate the initial car status to the system.
- The **car data update** is considered to bean easy complexity procedure as it has just to store data retrieved from the car when involved in a ride.

Procedure (EI)	**Complexity**	**Weight**
Registration request	High	06
Login	Low	03
Logout	Low	03
Start ride	Medium	04
Car data update	Low	03

External Outputs:

- The **password generation** is an average complexity operation as it provides a password to the user only under certain circumstances (to be assessed).
- The **car reservation** is a high complexity operation as it has to check that the user can reserve a car and open the session in case.
- The **reservation time-over** event is a complex procedure as it has to check whether the user does not reach the car in time and charge (s)he in case.
- The **ride cost forwarding** is an easy complexity procedure as it should just send the actual ride cost to the car that a user is riding.
- The **ride ending** is a complex procedure as it is supposed to close the session, assess the ride cost and get the user pay that cost.
- The **transaction handling** is an average complexity procedure as it is supposed to manage the transactions of the several users checking that those have been properly carried out.
- The **Management team calling** is an easy procedure as it has to request the Maintenance team intervention if the battery level of a car at the end of a session is too low.
- The **statistical monitoring** procedure is considered to be complex because it has to display a great amount of data grouped under certain criterion (Not shown in the RASD).

Procedure (EO)	**Complexity**	**Weight**
Password generation	Medium	05
Car reservation	High	07
Reservation time-over	High	07
Ride cost forwarding	Low	04
Ride ending	High	07
Transaction handling	Medium	05
Management team calling	Low	04
Statistical monitoring	High	07

External Inquiries:

- The **data checking procedure** is a pretty simple operation since it is just a query through an external server (then an input and output).
- The **profile info fetching** is an average complexity operation as it has to forward the account data to the user considering several data type.
- The **profile management** is an average complexity operation as it is supposed to let the user manage a great amount of data.
- The **car research** procedure is an average complexity operation as it has to select the available cars for a certain user under certain constraints.
- The **pledge checking** procedure is considered to be simple because it is supposed to query an external server and take care of the response.
- The **car unlocking** is a complex procedure as it has to unlock the car when the user requests it but assessing that certain requisites are submitted.
- The **MSO activation** is a complex procedure as it is has to suggest to the user a PGS near to the destination address that (s)he has inserted.
- The **car locking procedure** is a complex operation since that has to check the necessary condition to lock the car and eventually do that, it also has to check whether the car is plugged or not.
- The **administration intervention** is a group (6 as the main elements in the database) of several procedures considered of average complexity each. Here it is meant that an administrator can manage the data into the database while the system has to keep the database consistent.

Procedure (EQ)	**Complexity**	**Weight**
Data checking procedure	Low	03
Profile info fetching	Medium	04
Profile management	Medium	04
Car research	Medium	04
Pledge checking	Low	03
Car unlocking	High	06
MSO activating	High	06
Car locking	High	06
Administration intervention	Medium	24 = 04 * 6

Overall Estimation

Function type	**Function points**
Internal Logic Files	57
External Interface Files	0
External Inputs	19
External Outputs	46
External inquiries	60
Total	182

The total amount (in the table above) is yielded by summing up the function point values of the several function types and is used to estimate the logical **Lines Of Code** of the software taken under consideration just through the following formula:

$$\boldsymbol{LOC} = AVC \times TotFP = 46 \times 182 = 8372$$

Where:

- **AVC** is a constant value related to the chosen programming language (JEE then) considering the following link http://www.qsm.com/resources/function-point-languages-table
- **TotFP** is the total amount of Function Points assessed above

Now that the lines of code have been estimated, using COCOMO, the effort and cost to carry out the software development process is estimated.

3. Cocomo

***Note**: all the values and tables that follow are defined/chosen in the COCOMO model. Here the link to the reference guide:*
http://sunset.usc.edu/csse/affiliate/private/COCOMOII_2000_3/modelman.pdf

Following the COCOMO II model, starting from the LOC assessed, it is possible to reckon the effort in person/month (PM) by the following formula:

$$\boldsymbol{PM} = A \times KSLOC^{E} \times \prod_{i=1}^{n} EM_i$$

Where:

- **A** is a constant value (=2.94)
- **KSLOC** is a variable that express the Kilo Line Of Code previously estimated
- **E** is an aggregation of five **Scale Factors** (see below)
- **EM_i** is each one of the **n Effort Multipliers** that are derived from the **Cost Drivers** (see below)

Substituting the values, we got:

$$\boldsymbol{PM} = 2.94 \times 8372^{E} \times \prod_{i=1}^{n} EM_i$$

The **factor E** of the previous expression is figured out with the following formula:

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

Where:

- **B** is a constant factor (=0.91)
- **SF_j** is each of the five Scale Factor (see below)

Here the Scale Factor values are chosen (specified in parentheses) and the related motivations are given:

1. **PREC** (VL, 6.20): the very low band has been chosen because:
    - the goals of the project are of a pretty clear understanding
    - the actual team has never faced this kind of problem and therefore any form of experience has never been grown
    - No specific hardware is developed aside but the most functionalities are quite uncommon
    - It is likely that those uncommon functionalities have to be developed from scratch
2. **FLEX** (N, 3.04): the nominal band has been chosen because the project requirements are not that much strict but there are different external systems and communication protocols that the implemented system has to deal with. Moreover, no previous versions can be delivered in case of delay, therefore the completion time should be accurately estimated. Furthermore, nothing changes in case the project is completed early.

3. **RESL** (H, 2.83): the high band has been chosen because a good enough risk plan (and project planning) has been carried out and anyway the effort and cost estimation has been conducted in a slight pessimistically way.
   In detail, following the COCOMO table specified for this scale factor:
   - The Risk Management Plan identifies all critical risk items in a general way, establishes milestones for resolving them.
   - The schedule, budget, and internal milestones are generally compatible with Risk Management Plan.
   - About 20% of development schedule devoted to establishing architecture, given general product objectives.
   - Some support tool available for resolving risk items, developing and verifying architectural specs
   - Limited number and criticality of risk items
4. **TEAM** (H, 1.10): the high band has been chosen because the team can be considered to be highly cooperative.
   In detail, following the COCOMO table specified for this scale factor:
   - Strong consistency of stakeholder objectives and cultures
   - Full ability, willingness of stakeholders to accommodate other stakeholders' objectives
   - Considerable experience of stakeholders in operating as a team
   - Considerable stakeholder teambuilding to achieve shared vision and commitments

5. **PMAT** (VL, 7.80): the very low band has been chosen because following the CMMI parameters, considering that the company has just been found, the company is placed in the 1st (low) level.

Therefore, the **E factor** is:

$$\boldsymbol{E} = 0.91 + 0.01 \times (\,6.20 \;+\; 3.04 \;+\; 2.83 \;+\; 1.10 \;+\; 7.80\,) = \; 1.1197$$

With this, the effort formula becomes:

$$\boldsymbol{PM} = 2.94 \times 8.372^{1.1197} \times \prod_{i=1}^{n} EM_i$$

Now, to figure completely out the PM formula, the Effort Multipliers are assessed considering the Cost Drivers. Note that the project plan is being carried out in a kind of post-architecture phase, or to better say, between the RASD and DD delivery but having in mind how the system is supposed to end up.
Here the Cost Driver values are chosen, the relative grade is given in parentheses and the motivations directly follow:

- **Product Factors**
  1. **RELY** (H, 1.15): the system is supposed to be highly reliable since if that were down, the whole car sharing system would not be usable (as it is centralized)
  2. **DATA** (L, 0.94): the data are supposed to last 10 years in the database, then a backup is taken. Therefore, since for our service a storing of 100 Mbytes/years is needed, a 1 GB database is needed over 10 years. This leads that the rate Data Bytes / SLOC is equal to 119 and then the high band is chosen. Note that the

estimation is slightly pessimistic. However, this leads to convey that no big data has to be managed, then this field is set to a low score.

3. **CPLEX** (N, 1.00): the software is considered having a nominal complexity since being a quite large system but developed with an appropriate programming language that keeps a high level programming, leaving the details out, despite some functionalities could be considered pretty tricky.
   Going into the details, the assessed areas are:
   - control operations: H
   - computational operations: N
   - device-dependent operations: L
   - data management operations: N
   - user interface management operations: M

   that clearly leads to chose the overall nominal score band.
4. **RUSE** (N, 1.00): the reusability is limited to the project scope itself, therefore, the nominal band has been chosen.
5. **DOCU** (H, 1.08): the documentation thoroughly matches the life cycle needs, therefore, the high band has been chosen.

- **Platform Factors**
  6. **TIME** (H, 1.11): the execution time constraints over the software system are considered to behigh since many operations have to be carried out and someone are many of them are also time-constraining.
  7. **STOR** (N, 1.00): a normal storage constraint has been chosen since the hardware availability thoroughly covers the needs.
  8. **PVOL** (L, 0.87): the volatility constraint is set to low since the hardware and software is not supposed to be update so frequently despite the mobile applications that could require a slightly more maintenance.
- **Personnel Factors**
  9. **ACAP** (N, 1.00): although the problem analysis has been apparently conducted in a thorough way, the experience of the analyst is limited therefore the normal rate has been chosen.
  10. **PCAP** (N, 1.00): same as for the ACAP.
  11. **PCON** (H, 0.91): the project's annual personnel turnover is supposed to be very low therefore a high continuity is ensured.
  12. **AEXP** (L, 1.10): the team has the basis of the chosen programming language but no any complete system has been implemented so far. For this reason, the low score has been chosen.
  13. **PEXP** (L, 1.12): same as AEXP.
  14. **LTEX** (L, 1.11): same as PEXP.
- **Project Factors**
  15. **TOOL** (H, 0.88): a considerable amount of software tools (from coding to life-cycle support) are used then the high class has been selected.
  16. **SITE** (H, 0.85): the team is placed in the same city and presumably same building (beside being provided of good enough communication ways) therefore this parameter has been chosen as high.
- **General Factors**
  17. **SCED** (L, 1.08): low time constraints on the requirement scheduled are assumed then the low band has been chosen.

Here a summary table of the bands (and therefore the numerical values) chosen for each parameter.

Parameter (CD)	**Band**	**EM value**
RELY	H	1.15
DATA	L	0.94
CPLX	N	1.00
RUSE	N	1.00
DOCU	H	1.08
TIME	H	1.11
STOR	N	1.00
PVOL	L	0.87
ACAP	N	1.00
PCAP	N	1.00
PCON	H	0.91
AEXP	L	1.10
PEXP	L	1.12
LTEX	L	1.11
TOOL	H	0.88
SITE	H	0.85
SCED	L	1.08

Therefore, the productory of the Effort Multipliers is

$$\prod_{i=1}^{n} EM_i = 1.13342324$$

By this, the Person Months parameter is equal to:

$$\boldsymbol{PM} = 2.94 \times 8.372^{1.1197} \times \prod_{i=1}^{n} EM_i = \ 2.94 \times 8.372^{1.1197} \times 1.13342324 = 35.97753 \ \cong 36$$

Starting from the PM, considering that a person averagely works about 140 hours/month, the **person-hour effort** factor can be assessed in this way:

$$\boldsymbol{PH} = PM \times 160 = 36 \times 160 = 5040$$

Moreover, supposing a wage of 2000€/month gross (before taxes), we get a **total project cost** of:

$$\boldsymbol{TotalCost} = PM \times MediumWage = 36 \times 2000 \cong 72000 \text{ €}$$

However, once the PM is assessed, the **initial baseline schedule equation** is:

$$\boldsymbol{TDEV} = \left[3.67 \times PM^{\,(0.28 \,+\, 0.2 \,\times\, (B \,-1.01))}\right] \times \frac{SCED\%}{100}$$

Where:

- **PM** is the the effort expressed in Person Months Without the SCED multiplier (=33)

- **B** is the sum of the project scale factors (=0.91)
- **SCED%** is a factor depending on the SCED multiplier (=85%)

In the end, considering these values, the project duration indicator **TDEV is equal to 7.74 months**, amount that is supposed to be the calendar time in months from the determination of a product's requirements baseline to the completion of an acceptance activity certifying that the product satisfies its requirements.

4. Task Identification

The purpose of this section of the PPD is to identify the main tasks of each assignment seen as steps to be completed to redact the entire interested document.

- RASD:

Start date: 16-10-2016
Deadline: 19-11-2016

1. Goals
2. Assumptions and actors
3. Requirements
   - Functional
   - Non-functional
   - Goals-requirements-assumptions link table
4. Glossary and scenarios
5. UML models
   - Use cases models
   - Use case diagram
   - Class diagram
   - State-chart diagrams
   - Activity diagrams
   - Sequence diagrams
6. Mockups
7. Alloy

Precedence constraints:

- The *goals* task must begin before the *functional requirements* one.
- The *functional requirements* task must begin before the *UML models* one.
- The *goals-requirements-assumptions link table* task must begin after *goals, assumptions* and *requirements* one.
- The *mockups* task must begin after the *functional requirements* one.
- The *alloy* task must begin after *goals, assumptions* and *requirements* one.

- DD:

Start date: 25-11-2016
Deadline: 17-12-2016

1. Architectural design overview, styles and pattern
2. Component
   - Diagram
   - Description
   - Interfaces
   - Database logic view
3. Deployment view
4. Runtime view
5. Algorithm design
6. User interfaces design
   - UX diagram
   - BCE diagram
   - UX-component correspondence
7. Requirements traceability

Precedence constraints:

- The *requirements traceability* task must begin after the *component* one.
- The *user interfaces design* must begin after the *component* one.

- Development:

Start date: 07-01-2017
Deadline: 22-04-2017

1. Client side
2. Server side

- CI:

Start date: 30-04-2017
Deadline: 09-06-2017

- Unit testing

Start date: 30-04-2017
Deadline: 09-06-2017

- ITPD:

Start date: 10-06-2017
Deadline: 25-06-2017

- Integration testing

Start date: 26-06-2017
Deadline: 04-08-2017

- System testing

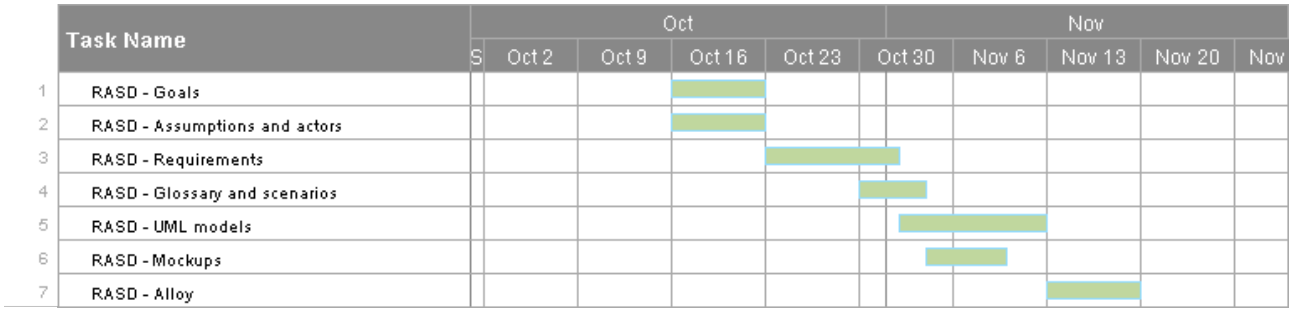
Start date: 05-08-2017
Deadline: 13-09-2017

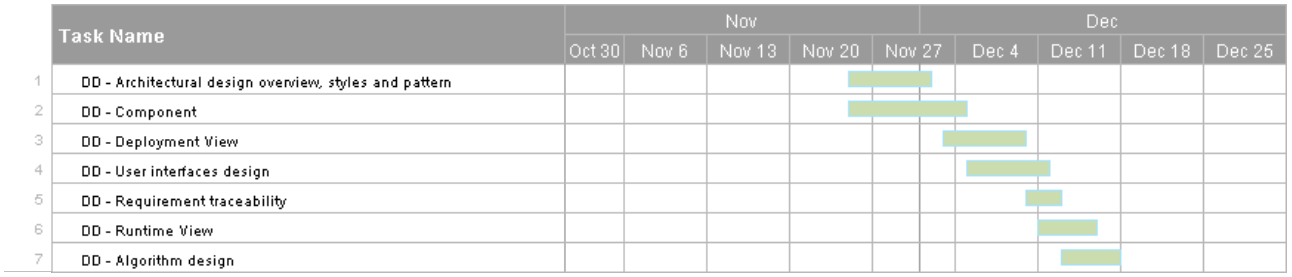
5. Schedule

In this section, we provide a general schedule of our tasks according to the estimation done in the first chapter. We've decided that the effective project plan takes place after the completion of the DD document, so we've split the RASD and DD's schedule from the other part of the project's one. The project plan includes also the development and testing part in order to simulate a truthful software producing.

5.1. RASD and DD schedule

After the completion of each document, the team is going to meet the stakeholder and discuss about the deliverables in order to apply possible changes (in requirements or in preferences) to be done before going on with the other task; in this way we try to prevent the risk of make changes about requirements or architecture after the development or testing (that is a very expensive problem).





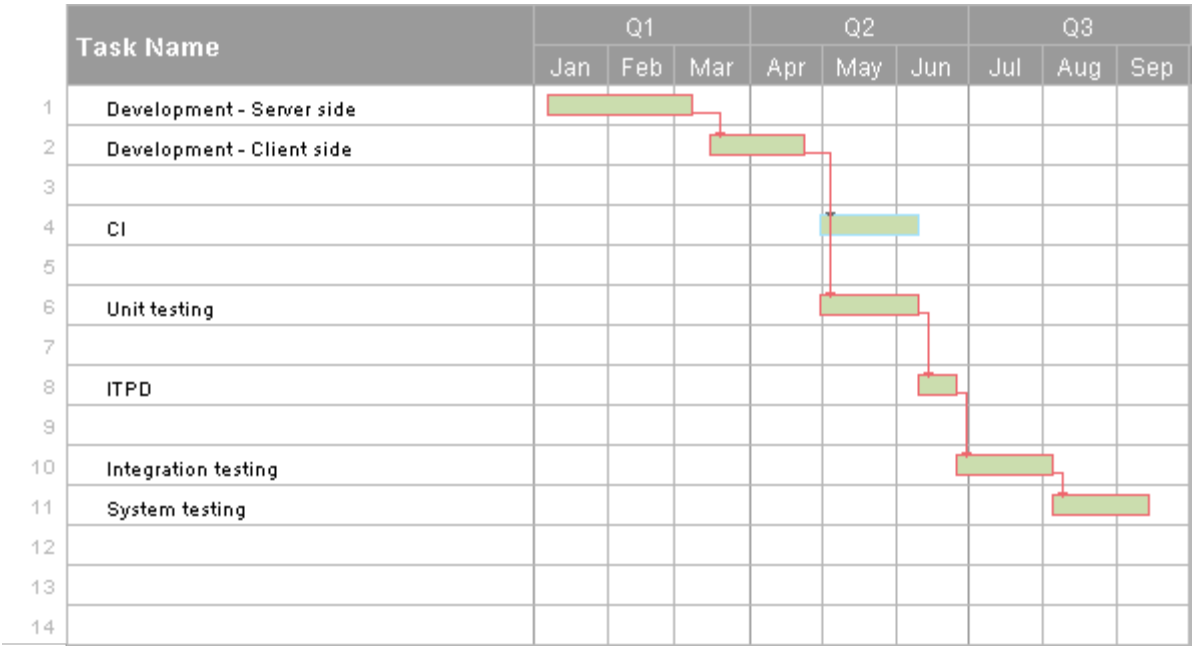
Deliveries:

RASD: 20-11-2016

DD: 18-12-2016

5.2. Project schedule

We assume that the days between the end of the DD (17-12-16) and 07-01-17 had been used to make the project plan, then the 07-01-17 the effective project took place. We've divided the diagram into trimesters to preserve readability, the total time from the development to the system testing takes 8 months during which there're break periods between a type of task and another to preserve some time for possible delays and for meetings with stakeholder.



Deliveries:

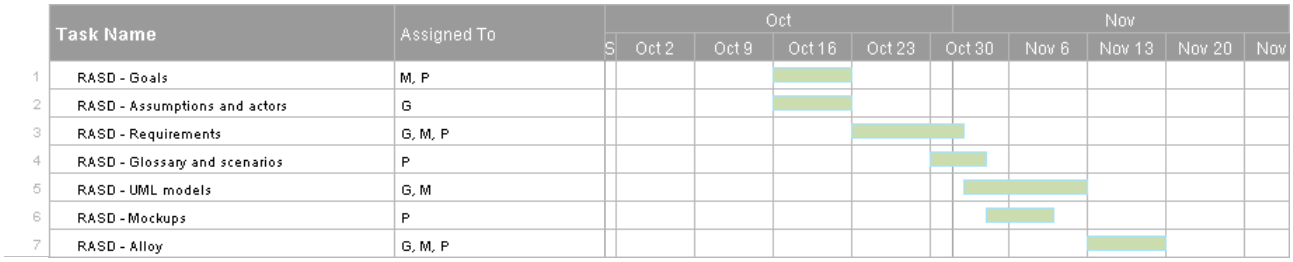
Development server side: 08-03-2017

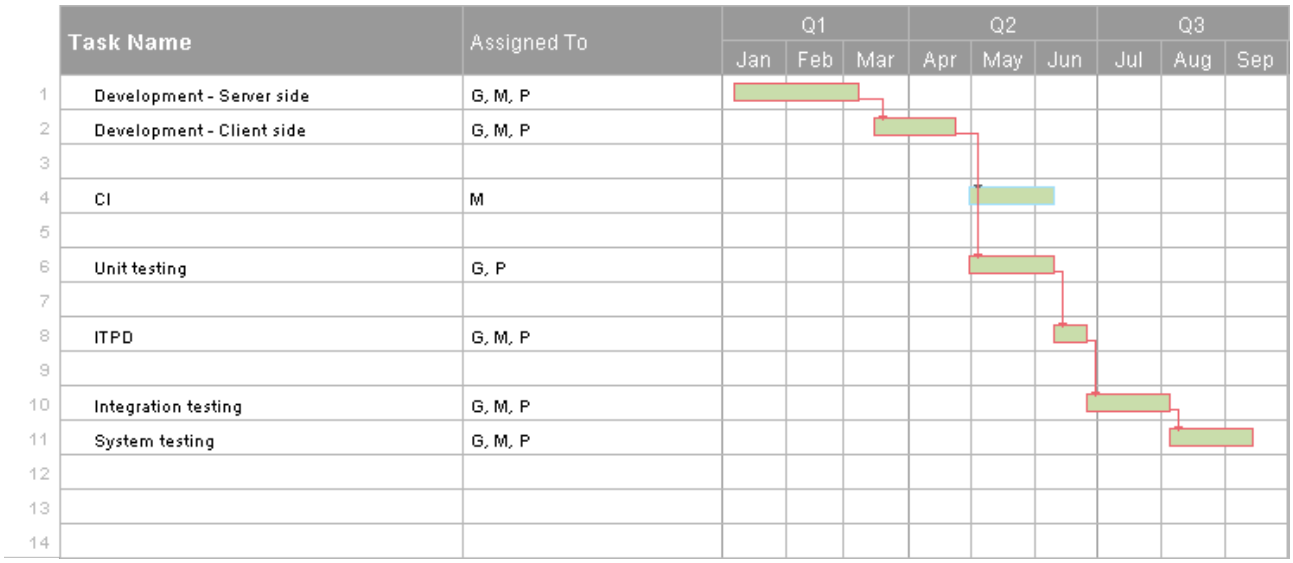
Development client side: 23-04-2017

Whole system: 14-09-2017

6. Resource Allocation

Here we present the scheduling allocation for each task. We've supposed that a person can be assigned to one task a time. In these diagrams where there're two overlapped activities, the person assigned to the latest activity leaves the older one (for example, in the RASD schedule the fourth task is overlapped to the third one, so when Paola starts to work on the fourth task, she leaves the third one to the other two team members).





7. Risk Management

In this document's section we're going to identify the possible risks that our project could go through during the realization. In addition to this, we want to classify them by relevance and suggest some ways to prevent them or, in other cases, to solve the issues they could create.
As long as possible we tried to build proactive risk strategies, but we adopt also the reactive risk strategy when it was more useful (if the proactive risk strategy was too expensive in terms of resources, time… or when the relevance of the risk was really low).

Scale (from less significant to the most):

- Probability:

- Effect:

Risk	It may happen that key staff could be ill for long periods of time or absent for other reasons
Probability	Moderate
Effect	Serious
Strategy	Organize the work to do in such a way that there's more than one person who knows how to deal with a specific task. Set frequently roundtables in which and encourage people to document their work as much as possible.

Risk	The schedule is too optimistic
Probability	Moderate
Effect	Serious
Strategy	When defining the schedule, foresee a few days with no other tasks than remedy the eventual delays

Risk	The estimated budget is too low
Probability	Low
Effect	Serious
Strategy	The budget has been computed in a "pessimistic" way in order to foresee eventual additional costs

Risk	The knowledge and skills of the staff are overrated
Probability	Medium
Effect	Serious
Strategy	Encourage the team work and provide external support, if really needed

Risk	A change in the requirements
Probability	High
Effect	Serious
Strategy	Create a requirement traceability, in order to be able to estimate the potential damage of that change and place the focus of that.

Risk	Problems with Maintenance Team
Probability	Moderate
Effect	Moderate
Strategy	Draw up a consistent and clear contract. If the problems remain, ask to someone else

Risk	Problems with Data Checking System
Probability	Low
Effect	Serious
Strategy	Draw up a consistent and clear contract. As it isn't a common service, it is difficult to replace

Risk	Problems with Payment System
Probability	Low
Effect	Moderate
Strategy	Draw up a consistent and clear contract. If the problems remain, look for different solutions

Risk	Problems with "off the shelf" component (such as navigation tools, etc)
Probability	Low
Effect	Moderate
Strategy	For those components that can be easily replaced, do so; otherwise study a new solution with the service's provider

Risk	Capability's problem in any (software) component of the system
Probability	Moderate
Effect	Serious
Strategy	Check frequently if the effective flow is similar to the expected one (see the statistical monitoring procedure), in order to be able to intervene timely

Risk	The cars available are not enough for the user's requests
Probability	Low (in the near future)
Effect	Moderate
Strategy	Draw up a flexible contract with the car's company

Risk	Problems due to the other car sharing companies' competition
Probability	Low
Effect	Moderate
Strategy	Study the market, find a market share and think of marketing strategies to be as competitive as possible

Risk	Problems with the stakeholders (We identify as stakeholders the company that has invested in the PowerEnJoy project)
Probability	Moderate
Effect	Serious
Strategy	Document in an accurate way the work done and future, in order to give reason of the costs

Risk	Complaints from the clients
Probability	Moderate
Effect	Moderate
Strategy	Institute a help service to collect data from the complaints

Risk	Loss of source code
Probability	Low
Effect	Catastrophic
Strategy	Use backups and version control software

8. Time effort

Each team member spent about 17 hours on the completion of this assignment.