Project Plan Document - v1.0

Gianpaolo Branca Luca Butera Andrea Cini

POLITECNICO
MILANO 1863

Contents

1 Introduction

1.1 Purpose and Scope

The purpose of this document is to provide an effective plan for the the realization of the PowerEnjoy service. We will perform a function points analysis to estimate the size of the project and then use the COCOMO II approach to produce a conservative estimation of the the cost and the effort the the project will require. We will then use the results of the analysis to produce a schedule compatible with the project size. Finally we will analyze the risks and the problems that could occur during the development cycles.

1.2 Definitions, Acronyms, Abbreviations

- Data element type (DET): a unique user recognizable, non recursive, field.
- Record element type (RET): Record elements type, a user recognizable subgroup of data elements.
- File types referenced (FRT): files updated or referenced in a transaction.
- MSO: Money saving option.
- SA: Safe Areas.

1.3 Reference documents

- The specification document.
- Project Plane document sample on the Beep platform.
- COCOMO II tables from: *sunset.usc.edu/research/COCOMOII/expert_cocomo/drivers.html*
- Function points reference: *www.functionpointmodeler.com/fpm-infocenter/index.jsp*

2 Project size, cost and effort estimation

In this section we will use well known approaches to project planning to estimate the dimension and the cost in time and money that our project will have. We will use the Function Points approach for the size estimation and than COCOMO for the cost and effort estimation and we will not consider, for the most part, the presentation layer.

2.1 Size estimation: function points

In the following tables the reference tables we are going to use for the size estimation, they classify the complexity of each element counting the numbers

of file types referenced, data element types and record element types. Obviously we are going to consider these reference values, but we are also going to estimate the complexity of the components based on the specific knowledge that we have acquired about the domain of our system.

Table 1: Logic Files

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

Table 2: My caption

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

Table 3: External Input

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

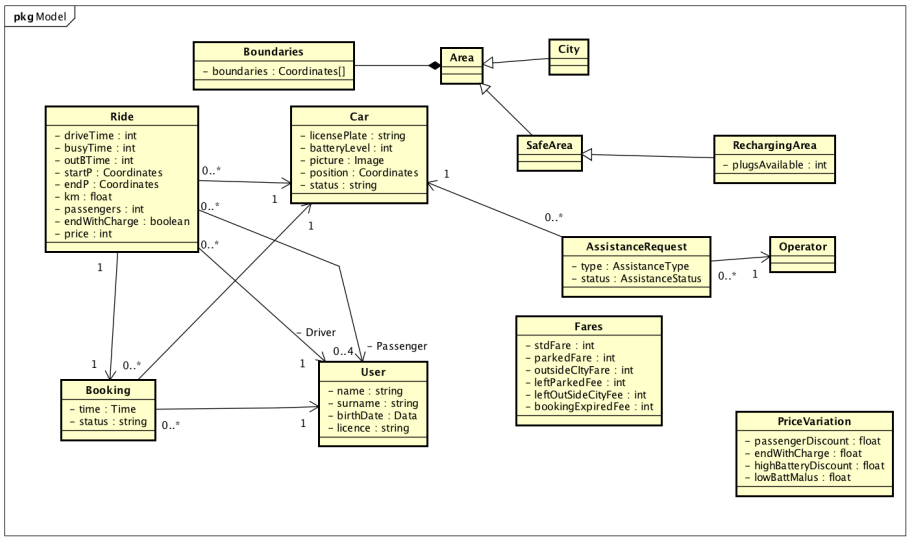
Table 4: Function Points

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

2.1.1 Internal Logic Files (ILFs)

In this section we are going to analyze the complexity of our ILFs, we are going to refer to this simplified version of the model of the internal representation

of our data (some attributes of the classes in the pictures are obviously not persistent but representative of the complexity of the informations related with the single entity):



- Users:
  - **Estimated complexity**: Average
    The first type of internal data our system will have to deal with are the user related ones. The handling of the user data, even if well standardized, requires a lot of care and because of the personalization functionality of the user's profile and periodic checks to be performed on driving licenses and PayPal accounts they will not be entirely static.
- Rides:
  - **Estimated complexity**: Low
    The rides data handling will be straight forward with the creation of the entity at the start of a ride and the update of the main fields when the ride is ended. The only dynamic aspect of the ride is the "paid" field that has to be updated when the payment is obtained.
- Bookings:
  - **Estimated complexity**: Low
    The information about cars bookings are static and easy to manage.
- Cars:
  - **Estimated complexity**: High
    The status of the cars are is the most critical type of data that our system will have to handle, informations about the cars dynamic and complex to retrieve.
- Assistance requests:

  - **Estimated complexity**: Low
    Easy informations to handle, static once generated(only the status changes only once).
- Fares/PriceVariations:
  - **Estimated complexity**: Low
    Static informations that can be updated by the management system, easy to handle and maintain.
- Safe and RechargingArea:
  - **Estimated complexity**: Average
    Static information about the position of safe areas and dynamic updated to the number of plugs available, average complexity (high complexity only for the creation of the objects)

Table 5: ILF

ILF	Complexity	FPs
Users Data	Average	10
Rides Data	Low	7
Bookings Data	Low	7
Cars Data	High	15
Assistance Requests Data	Low	7
System parameters	Low	7
Areas Data	Low	10
Total		63

2.1.2 External Logic Files (ELFs)

The external data sources we rely on are PayPal, for the payments handling and GoogleMaps for the maps and navigation related services.

- PayPal:
  - **Estimated complexity**: Low

  Simple files containing payment informations.
- GoogleMaps:
  - **Estimated complexity**: High

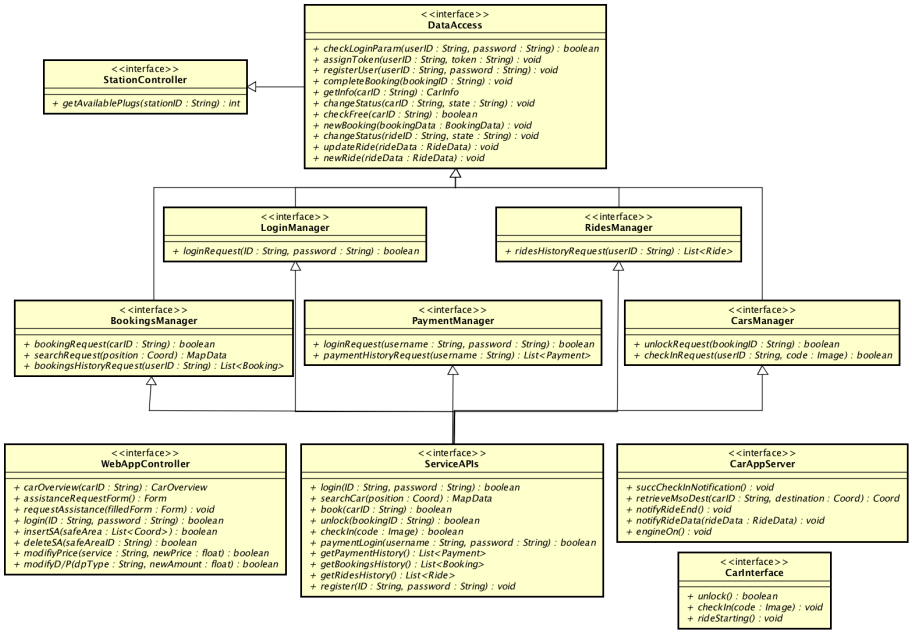
  We heavily rely on the files produced by the GoogleMaps service, for both the navigation and the maps functionalities, we consider that both of this elements have to be considered to have an high complexity.

Table 6: ELF

ELF	Complexity	FPs
Payment info	(Very)Low	2
Navigation Data	High	10
Maps Data	High	10
Total		22

2.1.3 External Inputs (EIs)

We report here the component interfaces diagrams that will become handy in the next sections.



Our system will have to deal mainly with the following kinds of inputs:

- Login/logout of operators and users: simple operations. Both low complexity.

From the users:

- Login/logout operation: simple operations with simple operations of validations on data. Both operations have a low complexity.
- Registration: operation of an average complexity, it requires only a small amount of checks.
- QR-code check-in: simple operation but that requires some components to be handled. Average complexity.
- car locking/unlocking: simple operations that involve a good number of components. Both average complexity.
- Search for car: complex operation that needs the collaboration of a good number of components. High complexity.
- Book/un-book a car: simple operations but that change the status of car. Both average complexity.
- Payment: simple operation. Low complexity.
- Money saving option: a complex operation that involves multiple components. High complexity.

- Ride data: data that the system receive during the ride. Hard to handle, high complexity.

From the operator:

- Get car overview: complex operation of data retrieval. High complexity.
- Assistance request: not so complex operation that generates a request form to be sent to the legacy system. Average complexity.
- Insertion/delation of safe or recharging areas: they are crucial parameters for the system. Both high complexity.
- Insertion/delation of a car: complex operations involving multiple components. Both high complexity.
- Overall view: overall view of all the cars and their position. High complexity.
- Parameters modification: simple modification but they have to be notified to the users. Average complexity.

Table 7: EIS

EIs	Complexity	FPs
Login/Logout	Low	2*3
Registration	Average	4
Qr-code check-in	Average	4
Car locking/unlocking	Average	2*4
Search for a car	High	6
Book/cancel	Average	2*4
Payment	Low	3
MSO	High	6
Ride data	High	6
Overall view	High	6
Car overview	High	6
Assistance request	Average	4
Insert/delete SA	High	2*6
Insert/delete car	High	2*6
Parameters modification	Average	4
Total		95

2.1.4 External Inquiries (EQs)

The main operations of simple data retrieval are:

- User profile: operation of average complexity.
- System parameters and safe area: both operation of average complexity.
- Payment history: low complexity operation, simple interaction with PayPal.
- Ride history: low complexity operation.

Table 8: EQ

EIs	Complexity	FPs
User Profile	Average	4
Payment history	Low	3
Ride history	Low	3
System parameters and SA	Average	2*4
Total		18

2.1.5 External Outputs (EOs)

The external output that our system produce are the mainly correlated with the notification system that we estimate having a cost of 12 function points. The other output is the assistance request sent to the legacy system, operation that has an average complexity.

Table 9: EOS

Eos	Complexity	FPs
Notification system	—	12
Assistance request	Average	5
Total		17

2.1.6 Overall Estimation

Our analysis has produced the following data:

Table 10: Total function points

Function Type	Value
Internal Logic Files	63
External Logic Files	22
External Inputs	95
External Inquiries	18
External Outputs	17
Total	215

With Java Enterprise edition as development platform and without considering in depth the aspects of the development of the mobile and web applications, we end up with the following results:

Average estimation: SLOC = 46 * 215 = **9890**

High estimation: SLOC = 67 * 215 = **14405**

(Function point language table from *www.qsm.com*)

2.2 Cost and effort estimation: COCOMO II

In this section we are going to use the COCOMO II model to estimate the cost and effort needed for our project.

2.2.1 Scale drivers

Table 11: Scale factors values

SFs	Very Low	Low	Nominal	High	Very High	Extra high
PREC	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	thoroughly familiar
SFj	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
SFj	5.07	4.05	3.04	2.03	1.01	0.00
TEAM	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
SFj	5.48	4.38	3.29	2.19	1.10	0.00
RESL	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
Sfj	7.07	5.65	4.24	2.83	1.41	0.00
PMAT	Level 1 Lower	Level 1 Upper	Level 2	Level 3	Level 4	Level 5
SFj	7.80	6.24	4.68	3.12	1.56	0.00

Brief description of the Scale Factors and motivation for our choices:

- **Precedentedness**: reflects the experience of the development team in similar projects. Since our team has no experience in developing large web applications in our case will be Low.
- **Flex**: reflects the degree of freedom in the development process in relation to constraints in requirements and external interfaces to use. In our case the requirements are quite strict, but we have a good degree of choice from an implementation point and in some functionalities. We think Nominal it's the correct value.
- **Team**: reflects the level of cooperation between the team members. In our case the correct value is High.

- **Risk resolution**: reflects how good the architecture as been defined and if risks has been taken into account. Since we are not expert in designing systems like this one, even if we have a good risk plan, to be conservative, we think that Nominal is the right value.
- **Process maturity**: after problems faced during the the first phases of the project have been overcome, we are q confident of our team capabilities even if we are unexperienced, but it's better to be conservative even in this case. The correct value is Nominal.

Table 12: Scale drivers values

Scale Driver	Factor	Value
Precedentedness (PREC)	Very low	6.20
Development flexibility (FLEX)	Nominal	3.04
Team cohesion (TEAM)	High	2.19
Risk resolution (RESL)	Nominal	4.24
Process maturity (PMAT)	Nominal	4.68
Total		20.35

2.2.2 Cost drivers

- Required Software reliability

  Reliability issues in our system could lead to the loss of resources, but as long as we are quick in detecting the issues relying on a assistance service provided by the legacy system could mitigate the losses. The correct value for the RELY cost driver is Nominal.

Table 13: RELY

RELY Cost Driver						
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 multipliers	0.82	0.92	1.00	1.10	1.26	n/a

- Database size:

  We expect the effective size of out database to be high in relation to the expected SLOC. An High value is probably the most sensible choice.

Table 14: DATA

DATA Cost Driver						
DATA Descriptors		D/P < 10	10 <D/P < 100	100 <D/P <1000	D/P > 1000	
Rating level	Ver Low	Low	Nominal	High	Very High	Extra high
Effort multipliers	n/a	0.90	1.00	1.14	1.28	n/a

- Required reusability:

  The software produced in our project could be reusable for a car sharing service of a larger scale. The right RUSE cost driver is High.

Table 15: RUSE

RUSE Cost Driver						
RUSE Descriptors		None	Access project	Across program	Across product line	Across multiple product line
Rating level	Ver Low	Low	Nominal	High	Very High	Extra high
Effort multipliers	n/a	0.95	1.00	1.07	1.15	1.24

- Documentation match to life-cycle needs

  The documentation in our project covers the life cycle need of the system with care. The value is set to Nominal.

Table 16: DOCU

DOCU Cost Driver						
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	Ver Low	Low	Nominal	High	Very High	Extra high
Effort multipliers	0.81	0.91	1.00	1.11	1.23	n/a

- Execution time constraint:

  Since we will have part of the logic distributed over cars, the workload of the system should not be too heavy. TIME is set to Nominal.

Table 17: TIME

TIME Cost Driver						
TIME Descriptors			<50%use of available execution time	70% use of available execution time	85% use of available execution time	95% use of available execution time
Rating level	Ver Low	Low	Nominal	High	Very High	Extra high
Effort multipliers	n/a	n/a	1.00	1.11	1.29	1.63

- Storage constraint:

  Even if our system will have to deal with a reasonable amount of data, we do not expect problems related to the storage availability. STOR is set to Nominal.

Table 18: STOR

STOR Cost Driver						
STOR Descriptors			<50%use of available storage	70% use of available storage	85% use of available storage	95% use of available storage
Rating level	Ver Low	Low	Nominal	High	Very High	Extra high
Effort multipliers	n/a	n/a	1.00	1.05	1.17	1.46

- Platform Volatility:

  We do not expect the system to be under constant evolution. Nominal is the right value for the PVOL cost driver.

Table 19: PVOL

PVOL Cost Driver						
PVOL Descriptors		Major change e. 12mo. Minor change e. mo.	Major change e. 6mo Minor change e. 2wk.	Major change e. 2mo Minor change e. 1wk.	Major change e. 2wk Minor change e. 2 days	
Rating level	Ver Low	Low	Nominal	High	Very High	Extra high
Effort multipliers	n/a	0.87	1.00	1.15	1.30	n/a

- Analyst Capability:

We are confident to have done a good work in analyzing the domain of our system. The ACAP driver is set to High.

Table 20: ACAP

ACAP Cost Driver						
ACAP Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating level	Ver Low	Low	Nominal	High	Very High	Extra high
Effort multipliers	1.42	1.19	1.00	0.85	0.71	n/a

- Programmer Capability:

  We are confident in our team ability in programming. PCAP is set to high.

Table 21: PCAP

PCAP Cost Driver						
PCAP Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating level	Ver Low	Low	Nominal	High	Very High	Extra high
Effort multipliers	1.34	1.15	1.00	0.88	0.76	n/a

- Application experience:

  Even if we are experienced in Java, this is our first JEE application, so to be conservative, APEX is set to Low.

Table 22: APEX

APEX Cost Driver						
Apex Descriptors	<2 months	6 months	1 years	3 years	6 years	
Rating level	Very Low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.22	1.10	1.00	0.88	0.81	n/a

- Platform Experience:

  As already said, we are not experienced in JEE and web development, PLEX is set to Low.

Table 23: PLEX

PLEX Cost Driver						
PLEXDescriptors	<2 months	6 months	1 year	3 years	6 years	
Rating level	Ver Low	Low	Nominal	High	Very High	Extra high
Effort multipliers	1.19	1.09	1.00	0.91	0.85	n/a

- Language and Tool experience:

  We are not familiar with the kind of development tools needed for our project. LTEX is set to Low.

Table 24: LTEX

LTEX Cost Driver						
LTEX Descriptors	<2 months	6 months	1 year	3 years	6 years	
Rating level	Ver Low	Low	Nominal	High	Very High	Extra high
Effort multipliers	1.20	1.09	1.00	0.91	0.84	n/a

- Personnel continuity:

  We don't expect an high turnover in our small team. PCON is set to High.

Table 25: PCON

PCON Cost Driver						
PCON Descriptors	48% year	24% year	12% year	6% year	3% year	
Rating level	Ver Low	Low	Nominal	High	Very High	Extra high
Effort multipliers	1.29	1.12	1.00	0.90	0.81	n/a

- Usage of Software Tools:

  We have well documented the tools that we are going to use in development, we think the right value for TOOL is High.

Table 26: TOOL

TOOL Cost Driver						
TOOL Descriptors	edit, code, debug	simple, frontend, backend, CASE, little integration	basic life-cycle tools, moderately integrated	strong, mature life-cycle tools, moderately integrated	string, mature, proactive life-cycle tools, well integrated with processes, methods, reuse	
Rating level	Ver Low	Low	Nominal	High	Very High	Extra high
Effort multipliers	1.17	1.09	1.00	0.90	0.78	n/a

- Multisite development:

  We will work in the same city with frequent meetings and simplicity in communications. SITE is set to High.

Table 27: SITE

SITE Cost Driver						
SITE Collocation Descriptors	International	Multi-city or multi-company	Multi-city or multi-company	Same city or metro area	Same building or complex	Fully collocated
SITE Communications Descriptors	Some phone, mail	Individual phone, fax	Narrow band email	Wideband electronic comm.	Wideband electr. comm., occasional video conf.	Interactive multimedia
Rating level	Ver Low	Low	Nominal	High	Very High	Extra high
Effort multipliers	1.22	1.09	1.00	0.93	0.86	0.80

- Required development schedule:

  The schedule for our project was not particularly stretched out. SCED is set to Nominal.

Table 28: SCED

SCED Cost Driver						
SCED descriptors	75% of nominal	85% of nominal	100% of nominal	130% of nominal	160% of nominal	
Rating level	Ver Low	Low	Nominal	High	Very High	Extra high
Effort multipliers	1.43	1.14	1.00	1.00	1.00	n/a

- Product complexity:

  Referencing to the COCOMO II classification, we think that the right value fort the complexity of our system is High.

Table 29: CPLEX

CPLX Cost Driver						
Rating level	Ver Low	Low	Nominal	High	Very High	Extra high
Effort multipliers	0.73	0.87	1.00	1.17	1.34	1.74

Overall results:

Table 30: Results

Cost Driver	Factor	Value
Required Software Reliability (RELY)	Nominal	1.00
Database size (DATA)	High	1.14
Required Reusability (RUSE)	High	1.07
Documentation match to life-cycle needs (DOCU)	Nominal	1.00
Execution Time Constraint (TIME)	Nominal	1.00
Main storage constraint (STOR)	Nominal	1.00
Platform volatility (PVOL)	Nominal	1.00
Analyst capability (ACAP)	High	0.85
Programmer capability (PCAP)	High	0.88
Application Experience (APEX)	Low	1.15
Platform Experience (PLEX)	Low	1.09
Language and Tool Experience (LTEX)	Low	1.09
Personnel continuity (PCON)	High	0.90
Usage of Software Tools (TOOL)	High	0.90
Multisite development schedule (SITE)	High	0.93
Required development schedule (SCED)	Nominal	1.00
Product complexity (CPLEX)	High	1.17
Total (Product)		1.09874

2.2.3 Effort equation

Finally we can use the values produced by our analysis to get the final estimation in PM (Person Months). (Simple calculations in Python)

```
A = 2.94
EAF = 1.09874
SF = [6.20, 3.04,  2.19, 4.24, 4.68]
KSLOC_avg = 9890.0 / 1000.0
KSLOC_max = 14405.0 / 1000.0
B = 0.91
E = B + 0.01 * sum(SF)

effort_avg = A * EAF * KSLOC_avg ** E #PM
effort_max = A * EAF * KSLOC_max ** E #PM
```

With results:

```
>>>effort_avg
41.43748003927392

>>>effort_max
62.9863995508225
```

2.2.4 Schedule estimation

The estimation of the duration of the project is:

```
F = 0.28 + 0.3 * (E - B)
duration_avg = 3.67 * effort_avg ** F #months
duration_max = 3.67 * effort_max ** F #months
```

With results:

```
>>>duration_avg
13.070074457743738

>>>duration_max
15.076453059081716
```

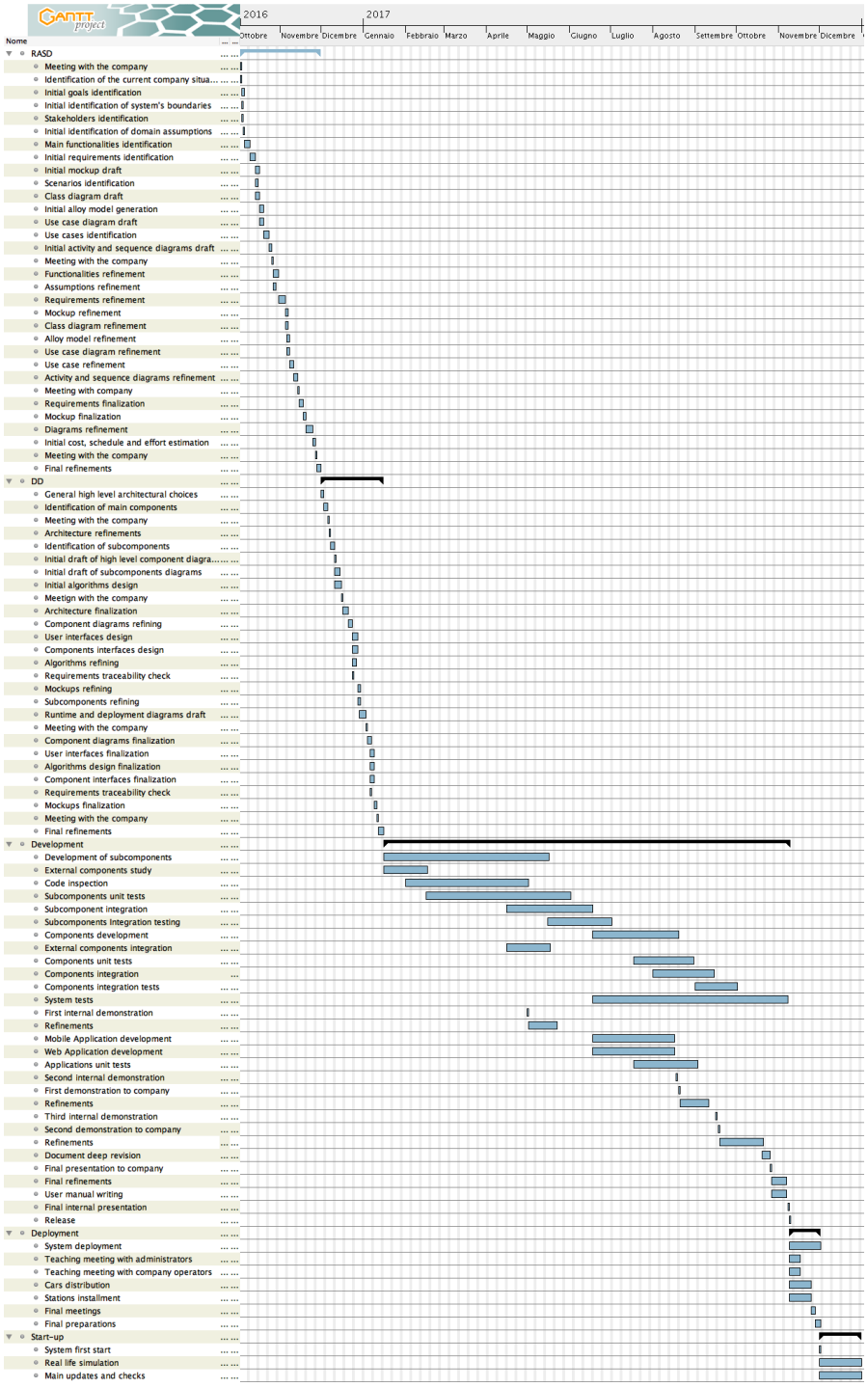
We will use these values in the next sections to produce a proper schedule for our project.

3 Schedule

In this section an initial high level schedule is presented in the form of a bar chart. The schedule will likely be updated, as the project goes on, to better fit

the situations that may occur. We provided a schedule containing also phases that aren't actually part of this project due to its didactical purpose, for example we aren't going to perform any development, but we found practical and useful to show how an actual scheduling (in a real situation) would have been for a project of this size.ù

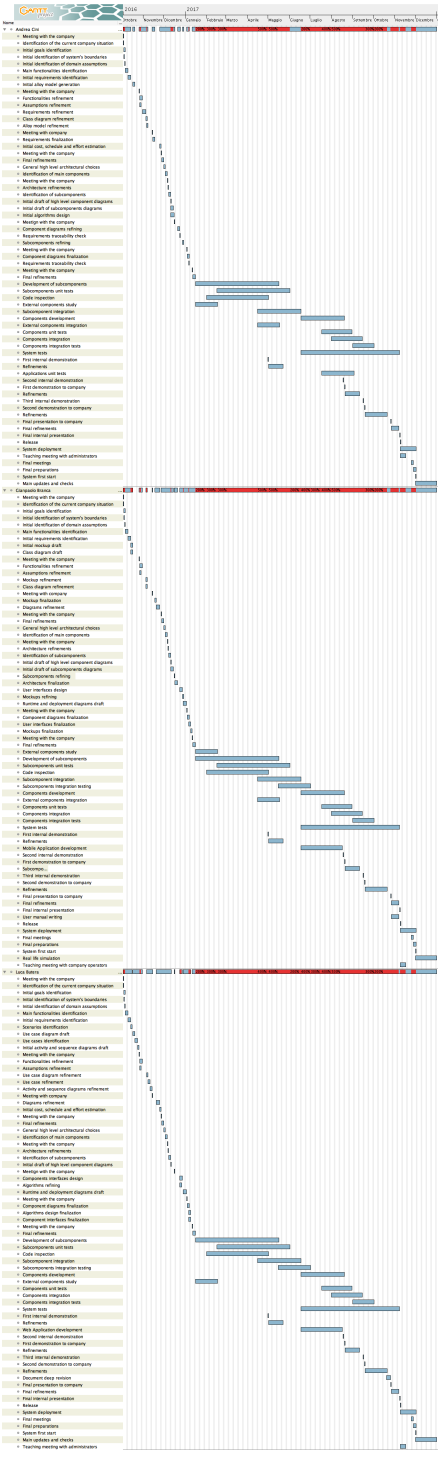
(Full size image here *https://raw.githubusercontent.com/GianpaoloBranca/SoftwareEng2/master/scratches/PPD/images/Schedule_bar_chart.png*)


GANTT project
2016
2017
Ottobre
Novembre
Dicembre
Gennaio
Febbraio
Marzo
Aprile
Maggio
Giugno
Luglio
Agosto
Settembre
Ottobre
Novembre
Dicembre
Name
RASD
Meeting with the company
Identification of the current company situa...
Initial goals identification
Initial identification of system's boundaries
Stakeholders identification
Initial identification of domain assumptions
Main functionalities identification
Initial requirements identification
Initial mockup draft
Scenarios identification
Class diagram draft
Initial alloy model generation
Use case diagram draft
Use cases identification
Initial activity and sequence diagrams draft
Meeting with the company
Functionalities refinement
Assumptions refinement
Requirements refinement
Mockup refinement
Class diagram refinement
Alloy model refinement
Use case diagram refinement
Use case refinement
Activity and sequence diagrams refinement
Meeting with company
Requirements finalization
Mockup finalization
Diagrams refinement
Initial cost, schedule and effort estimation
Meeting with the company
Final refinements
DD
General high level architectural choices
Identification of main components
Meeting with the company
Architecture refinements
Identification of subcomponents
Initial draft of high level component diagra...
Initial draft of subcomponents diagrams
Initial algorithms design
Meetign with the company
Architecture finalization
Component diagrams refining
User interfaces design
Components interfaces design
Algorithms refining
Requirements traceability check
Mockups refining
Subcomponents refining
Runtime and deployment diagrams draft
Meeting with the company
Component diagrams finalization
User interfaces finalization
Algorithms design finalization
Component interfaces finalization
Requirements traceability check
Mockups finalization
Meeting with the company
Final refinements
Development
Development of subcomponents
External components study
Code inspection
Subcomponents unit tests
Subcomponent integration
Subcomponents integration testing
Components development
External components integration
Components unit tests
Components integration
Components integration tests
System tests
First internal demonstration
Refinements
Mobile Application development
Web Application development
Applications unit tests
Second internal demonstration
First demonstration to company
Refinements
Third internal demonstration
Second demonstration to company
Refinements
Document deep revision
Final presentation to company
Final refinements
User manual writing
Final internal presentation
Release
Deployment
System deployment
Teaching meeting with administrators
Teaching meeting with company operators
Cars distribution
Stations installment
Final meetings
Final preparations
Start-up
System first start
Real life simulation
Main updates and checks

4 Resource allocation

In this section we split the tasks, defined in the previous scheduling section, among the different components of the project team. As previously stated, the Resource allocation bar chart is likely to be updated as the project progresses, also to better define the granularity of tasks division among team members. We are aware that, specially in the development section of the chart, tasks are not too much divided among members, this due, firstly, to the low number of "workers" involved in this project, that clearly doesn't fit a development process for such a system, secondly, to the height of this chart in terms of abstraction levels.

(Full image here *https://raw.githubusercontent.com/GianpaoloBranca/SoftwareEng2/master/scratches/PPD/images/Resources_bar_chart.png*)



5 Risk management

In this section, risks for our system development are taken into account, along with some possible strategies to tackle them effectively.

Problems at a management level are not so likely to happen since the project was commissioned by a company and there are contractual duties to satisfy even in case of a changes in the high spheres. Anyway, the high number of meetings should always keep our employers with an eye inside the process, so that they will consider the importance of the project itself for the company's future.

Along with managements problems are budget problems, they're mostly unpredictable and rather serious when they occur, the only strategy here is to check in advantage the financial situation of the company. In our case, since we work for a transportation company with an already solid bases, it's not likely that this kind of problems will occur.

For what concerns risks of requirements misunderstandings we tackle them in advance via the many encounters with the company managers, so that at various levels in the development process their correctness is checked. Even risks about external components are at a minimum, for the system can rely on solid ones that are not likely to exit the market or change in short time.

For what concerns the actual salability of the service, we are by no mean worried, the city in which the service will be implemented is a metropolis, therefore is unlikely that such a system will not get many users; a secondary problem may be the concurrency of similar services, but the system will tackle such a problem thanks to its flexibility in fares.

Some problems, which are more likely to happen over a 2 year course, are illness in the personal, especially if this cases take place during important milestones in the project or meetings, to address this kind of problems is useful to keep cohesion among team members at the level of comprehension of teammates work and to always have groups of people working at each task, so that the absence of one doesn't block the developing process. The problems regarding the staff can be many and different, in case of high specialization members, for example, their absence can seriously damage the project, therefore is safer to hire people with a wide range of skills, for the system we're going to develop doesn't advice the need for any particularly high specialized professional.

In any case, as a matter of precaution, the time for the various tasks in the scheduling bar chart, is slightly overestimated, so that in case of shifts in the project progress we're covered.

6 Effort spent

Each group member has spent around 10 hours working on this document.