Politecnico di Milano, A.A. 2016/2017

Software Engineering 2: PowerEnJoy
Project Plan

Binosi Lorenzo 876022

January 22, 2017

Contents

1 Introduction

1.1 Revision History

Version of this document: 1.0
Last update: x

1.2 Purpose

This document is the Project Plan Document (PPD) for the PowerEnJoy project. Its aim to estimate which are the sizes, the costs and the efforts that should be faced in order to realize the project. According to the estimation made, it will define the required budget, the resources allocation and the schedule of the activities.

The document is written for customers, project managers, developers, testers and all the stakeholders involved in the project implementation.

1.3 Scope

PowerEnJoy is a digital management system for a car-sharing service that exclusively employs electric car. This car rental system provides an alternative solution to public transport, thus being not only eco-friendly, but also simple and reliable. Its implementation should be achieved in a clear and defined way in order to meet the established deadlines, and avoid any further cost.

1.4 Definitions, acronyms, and abbreviations

1.5 Reference Documents

This document refers to the following documents:

- Software Engineering 2 project [1].
- PPD assignement [2].
- RASD of the PowerEnJoy project [3].
- DD of the PowerEnJoy project [4].
- ITPD of the PowerEnJoy project [5].
- COCOMO II - Model Definition Manual [6].

2 Project size, cost and effort estimation

This section describes which are the main functionalities of PowerEnJoy that allowed us to estimate the expected size, the cost and the required effort of the project.

For the size estimation we will use the **Function Points** approach. It allows to estimate the correspondent amount of lines of code to be written in Java, according to the main functionalities of PowerEnJoy. We won't consider the functionalies of the user interface in order to have a more meaningful and reliable estimation.

For the cost and effort estimation we will use the **COCOMO II** model, relying on the amount of lines of code estimated previously.

2.1 Size estimation

For the size estimation we will refer to the IFPUG 1994 standard [7] which specifies the definitions, rules and steps for applying the IFPUG's functional size measurement (FSM) method.

In order to determine the complexity level, the IFPUG standard classifies each function count into Low, Average and High complexity levels depending on the number of data element types contained and the number of file types referenced. Tables 2.1a, 2.1b and 2.1c reassume the complexity levels for each type of file.

Once determined the complexity of each function, it's possible to define its weight, i.e., the relative effort required to implement it. Table 2.2 reassume the Unadjusted Function Points (UFP) for each complexity level.

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

(a) Complexity estimation for ILFs and EIFs.

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

(b) Complexity estimation for EOs and EQs.

	Data Elements		
Record Elements	1-4	5-15	16+
1	Low	Low	Avg.
2-3	Low	Avg.	High
3+	Avg.	High	High

(c) Complexity estimation for EIs.

Table 2.1: Estimation of complexities for different types of Function Points.

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

Table 2.2: UFP complexity weights.

2.1.1 Internal Logic Files (ILFs)

The ILFs are those logical files generated, used or mantained by the software system. In our case they include all the information that the system has to store:

- Driver:
  - Driver information
  - CreditCard
- Rent:
  - Rental information
  - RentalEvents
- Car
- SafeArea
- CarAssistance

The Driver Record Element (RET) contains 15 Data Elements (DET's) and the CreditCard RET, which contains 2 DET's. Hence, according to the Table 2.1, the Driver ILF has a low complexity with an amount of 2 RET's and 17 DET's.

The Rent RET contains 6 DET's and the RentalEvent RET, which contains 2 DET's. Hence, the Rent ILF has a low complexity with an amount of 2 RET's and 8 DET's.

The Car, SafeArea and CarAssistance RET's contain respectively 11, 4 and 4 DET's, and their ILFs have a low complexity.

ILF	**Complexity**	**FPs**
Driver	Low	7
Rent	Low	7
Car	Low	7
SafeArea	Low	7
CarAssistance	Low	7
Total		**35**

Table 2.3: The ILFs complexity and the total Function Points.

2.1.2 External Interface Files (EIFs)

PowerEnJoy needs to interface with 3 services, described in the DD, in order to perform some operations. They are:

SMSGateway : for the SMS dispatching.

EmailSender : for the email dispatching.

PaymentGateway : for the payment execution.

Obviously, each of these services doesn't require too much data, so we can consider their complexity as low.

EIF	**Complexity**	**FPs**
SMSGateway	Low	5
EmailSender	Low	5
PaymentGateway	Low	5
Total		**15**

Table 2.4: The EIFs complexity and the total Function Points.

2.1.3 External Inputs (EIs) and External Inquiries (EQs)

The PowerEnJoy business logic provides a RESTful API for all the core functionalities of the system. All the API requests are completely described in the DD [2.5.3, p. 14] and summarized below. Thanks to this detailed description, the complexity estimation of the EIs and the EQs will be performed accurately.

- User creation: this request allows the creation of a new user. It requires all of the user information as input and doesn't provide any output data. According to Table 2.1c, the EI of the request has an high complexity with an amount of 3 RET's and 12 DET's.
- User's salt bytes retrieval: this request allows to retrieve the user's salt bytes in order to use them for the password hashing. It requires only the username of the user as input and provides the user's salt bytes as output. The EI and the EQ of the request have both a low complexity.
- User deletion: this request allows to delete the user that performs the request. It requires only the credentials of the user (username and

hashed password) as input and doesn't provide any output data. The EI of the request has a low complexity.

- User's rental logbook retrieval: this request allows a user to retrieve his/her rental logbook. It requires only the credentials of the user as input and provides the logbook information as output. The EI and the EQ of the request have both a low complexity. In particular the low complexity of the EQ is given by an amount of 2 RET's and 5 DET's, according to Table 2.1b.

- Available cars' retrieval: this request allows a user to retrieve all the available cars in the nearby area. It requires the credentials of the user and his/her geographical position as input, and provides the data about the available cars as output. The EI and the EQ of the request have both a low complexity.

- Car reservation: this request allows a user to reserve a choosen car. It requires the credentials of the user and the license plate of the car as input, and doesn't provide any output data. The EI of the request has a low complexity.

- Current reservation retrieval: this request allows a user to retrieve the information about the current reservation. It requires the credentials of the user as input and provides the information about the current reservation. The EI and the EQ of the request have both a low complexity.

- Unlocks car: this request allows a user to unlock a reserved car. It requires the credentials of the user and his/her position as input, and doesn't provide any output data. The EI of the request has a low complexity.

- Car heartbeat: this request allows any car to be tracked by the system. It requires all the information about the car status and rent, and doesn't provide any output data. The EI of the request has an high complexity with an amount of 6 RET's and 10 DET's.

- Available safe areas' retrieval: this request allows any car to provide all the available neaby safe areas to the driver. It requires the credentials of the car and its geographical position, and provides the information about the available nearby safe areas. The EI and the EQ of the request have both a low complexity.

EI	Complexity	FPs
User creation	High	6
User's salt bytes retrieval	Low	3
User deletion	Low	3
User's rental logbook retrieval	Low	3
Available cars' retrieval	Low	3
Car reservation	Low	3
Current reservation retrieval	Low	3
Unlocks car	Low	3
Car heartbeat	High	6
Available safe areas' retrieval	Low	3
Total		**36**

Table 2.5: The EIs complexity and the total Function Points.

EQ	Complexity	FPs
User's salt bytes retrieval	Low	3
User's rental logbook retrieval	Low	3
Available cars' retrieval	Low	3
Current reservation retrieval	Low	3
Available safe areas' retrieval	Low	3
Total		**15**

Table 2.6: The EQs complexity and the total Function Points.

2.1.4 External Outputs (EOs)

Sometimes the system has to notify the user outside the context of a response. These occasions are:

- Notify the user that his/her rent has been concluded.
- Notify the user that his/her rent cannot be concluded.
- Notify the user that the payment of the rent has been received.
- Notify the user that the payment of the rent hasn't been received.

Like the EIFs, each of these notification doesn't require too much data, so we can consider their complexity as low.

EO	Complexity	FPs
Rent concluded	Low	4
Rent not concluded	Low	4
Payment received	Low	4
Payment not received	Low	4
Total		**16**

Table 2.7: The EOs complexity and the total Function Points.

2.1.5 Overall estimation

The following table summarizes the results of our estimation activity:

Function Type	Value
Internal Logic Files	35
External Interface Files	15
External Inputs	36
External Outputs	16
External Inquiries	15
Total	**117**

Table 2.8: Count of UFPs of our system.

As already said, the estimation made doesn't concern the mobile applications, the web server and the car application. Hence, we can finally convert the UFPs in the total number of lines of Java code.

According to IFPUG, the multiplicator to convert the Unadjusted Function Points to Source Lines of Code for Java is 53.

$$\text{SLOC} = 117 * 53 = 6201$$

2.2 Cost and effort estimation

For the cost and the effort estimation we will use the COCOMO II approach. In order to get a feasibile estimation, we suppose that our team has just been made up and is formed by 3 people: Lorenzo, Bob and Alice.

2.2.1 Scale Factors

The scale factors are parameters that concern several aspects regarding the overall experience of our team and the fesibility of the given project. The following table summarize all the scale factors and their values, according to COCOMO II.

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	thoroughly 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 1 Lower	Level 1 Upper	Level 2	Level 3	Level 4	Level 5
SF_j	7.80	6.24	4.68	3.12	1.56	0.00

Let's explain what's the meaning of each factor and what could be the suitable values in our case.

Precedentedness: reflects the previous experience of our team with the development of large scale projects. Since this is the first project for our team, the value will be low.

Development flexibility: reflects the degree of flexibility in the development process. Since we declared many functional and non functional requirements in the RASD, the value will be low.

Risk resolution: reflects the level of awareness and reactiveness with respect to risks. The risk analysis we performed is quite extensive, so the value will be set to very high.

Team cohesion: reflects how well the development team know each other and work together. As we already said, this is the first project for our team, so the value will be low.

Process maturity: reflects the process maturity of the organization. Untill now, the project has been conducted correctly, so the value will be set to nominal.

The estimated scale factors for our project are shown in Table 2.10.

Code	**Name**	**Factor**	**Value**
PREC	Precedentedness	Low	4.96
FLEX	Development flexibility	Low	4.05
RESL	Risk resolution	Very High	1.41
TEAM	Team cohesion	Low	4.38
PMAT	Process maturity	Nominal	4.68
Total	$E = 0.91 + 0.01 \times \sum_i SF_i$		**1.1048**

Table 2.10: Scale factors for our project.

2.2.2 Cost Drivers

- Required software reliability:

  this is the measure of the extent to which the software must perform its intended function over a period of time. Since our system has been thought to provide a further and timely transport service, its reliability must be high.

RELY Cost Drivers						
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:

  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. We suppose to reach at least 5GB of bytes in the testing database, so the value of the D/P ratio will be about 800 (high).

DATA Cost Drivers						
DATA Descriptors		Testing DB bytes/pgm SLOC < 10	10 $\leq$D/P$\leq$ 100	100 $\leq$D/P$\leq$ 1000	DP > 1000	
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	0.90	1.00	1.14	1.28	n/a

- Product complexity:

  this cost driver attempts to capture the average complexity of the following areas: control operations, computational operations, device-dependent operations, data management operations and user interface management operations. Our estimation doesn't include the mobile applications, the web server and the car application. For this reason we are not able to estimate correctly this driver, hence we set its value to nominal.

CPLX Cost Driver						
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

- Required reusability:

  this cost driver accounts for the additional effort needed to construct components intended for reuse on current or future projects. This effort is consumed with creating more generic design of software, more elaborate documentation, and more extensive testing to ensure components are ready for use in other applications. Our project relies on many Java EE tools and components, and the coded part is strictly related to the project itself. Hence, the value will be low.

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

- Documentation match to life-cycle needs:

  this cost driver is evaluated in terms of the suitability of the projects documentation to its life-cycle needs. Considering the purpose of our system, its maintenance phase will be extremely important. For this reason, we have to provide an accurate documentation in order to minimize any further effort. Hence, the value of this driver will be very high.

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	Very 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:

  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. Even if the sofware of our system is not too complex, we are expecting a high amount of users which use our service every day. For this reason the value will be very high.

TIME Cost Driver						
TIME Descriptors			$\leq$ 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	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	n/a	1.00	1.11	1.29	1.63

- Storage constraint:

  this rating represents the degree of main storage constraint imposed on a software system or subsystem. In other words, it describes the expected amount of storage usage with respect to the availability of the hardware. Our system continuously store information about users and rents, so the usage percentage of the available storage will be extremely high.

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

- Platform volatility:

  for what concerns the core system, we don't expect our fundamental platforms to change very often. Hence, the value of this cost driver will be low.

PVOL Cost Driver						
PVOL Descriptors		Major change every 12 mo., minor change every 1 mo.	Major: 6mo; minor: 2wk.	Major: 2mo, minor: 1wk	Major: 2wk; minor: 2 days	
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	0.87	1.00	1.15	1.30	n/a

- Analyst capability:

  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. Being new to the development of large scale projects, the parameter is set to nominal.

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

- Programmer capability:

  this evaluation should be based on the capability of the programmers as a team rather than as individuals. Our team is composed by competent people which have never worked as team, so the value of this driver will be set to nominal.

PCAP Cost Driver						
PCAP Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	

Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.34	1.15	1.00	0.88	0.76	n/a

- Application experience:

  the rating for this cost driver is dependent on the level of applications experience of the project team developing the software system or subsystem. Our team have some experience in the development of Java applications, but haven't developed any Java EE applications yet. For this reason, the parameter will be set to low.

APEX Cost Driver						
APEX Descriptors	$\leq$ 2 months	6 months	1 year	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:

  our team have previous experiences with the development of database and servers, hence the parameter will be set to nominal.

PLEX Cost Driver						
PLEX Descriptors	$\leq$ 2 months	6 months	1 year	3 years	6 years	
Rating level	Very 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:

  our team have some experience in the development of Java applications, but haven't developed with Java EE tools yet. For this reason, the parameter will be set to nominal.

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

- Personnel continuity:

  this driver is evaluated in terms of the projects annual personnel turnover. Being this our first project, the value of this driver will be set to low.

PCON Cost Driver						
PCON Descriptors	48% / year	24% / year	12% / year	6% / year	3% / year	
Rating level	Very 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:

  Our application environment is complete and well integrated, so we'll set this parameter as high.

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	strong, mature, proactive life-cycle tools, well integrated with processes, methods, reuse	
Rating level	Very low	Low	Nominal	High	Very High	Extra High

Effort multipliers	1.17	1.09	1.00	0.90	0.78	n/a

- Multisite development:

  even if the people of our team lives in different cities, we have collaborated relying hugely on wideband Internet services including social networks and emails. For this reason, we're going to set this parameter to very high.

SITE Cost Driver						
SITE Collocation Descriptors	International	Multi-city and 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 communication	Wideband elect. comm., occasional video conf.	Interactive multimedia
Rating level	Very 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:

  this rating measures the schedule constraint imposed on the project team developing the software. Untill now, all the deadlines have been met, hence the parameter is set to nominal.

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

The estimated cost drivers for our project are shown in Table 2.28.

Code	Cost Driver	Factor	Value
RELY	Required software reliability	High	1.10
DATA	Database size	High	1.14
CPLX	Product complexity	Nominal	1.00
RUSE	Required reusability	Low	0.95
DOCU	Documentation match to life-cycle needs	Very High	1.23
TIME	Execution time constraint	Very High	1.29
STOR	Storage constraint	Extra High	1.46
PVOL	Platform volatility	Low	0.87
ACAP	Analyst capability	Nominal	1.00
PCAP	Programmer capability	Nominal	1.00
APEX	Application experience	Low	1.10
PLEX	Platform experience	Nominal	1.00
LTEX	Language and tool experience	Nominal	1.00
PCON	Personnel continuity	Low	1.12
TOOL	Usage of software tools	High	0.90
SITE	Multisite development	Very High	0.86
SCED	Required development schedul	Nominal	1.00
Total	$EAF = \prod_i C_i$		**2.2895**

Table 2.28: Cost drivers for our project.

2.2.3 Effort equation

This final equation gives us the effort estimation measured in Person-Months (PM):

```
Effort = A * EAF * KSLOC^E
```

where:

```
A = 2.94 (according to COCOMO II)
EAF = 2.2895
E = 1.1048
```

The total effort results in **50.5** person-months.

2.2.4 Schedule estimation

According to COCOMO II, the schedule estimation is given by the following formula:

```
Duration = 3.67 * Effort^F
```

where:

```
B = 0.91
F = 0.28 + 0.2 * (E − B) = 0.31896
```

The duration calculated results in a total of **12.82** months. The average number of people required for the development time is given by the following formula:

```
People = Effort / Duration
```

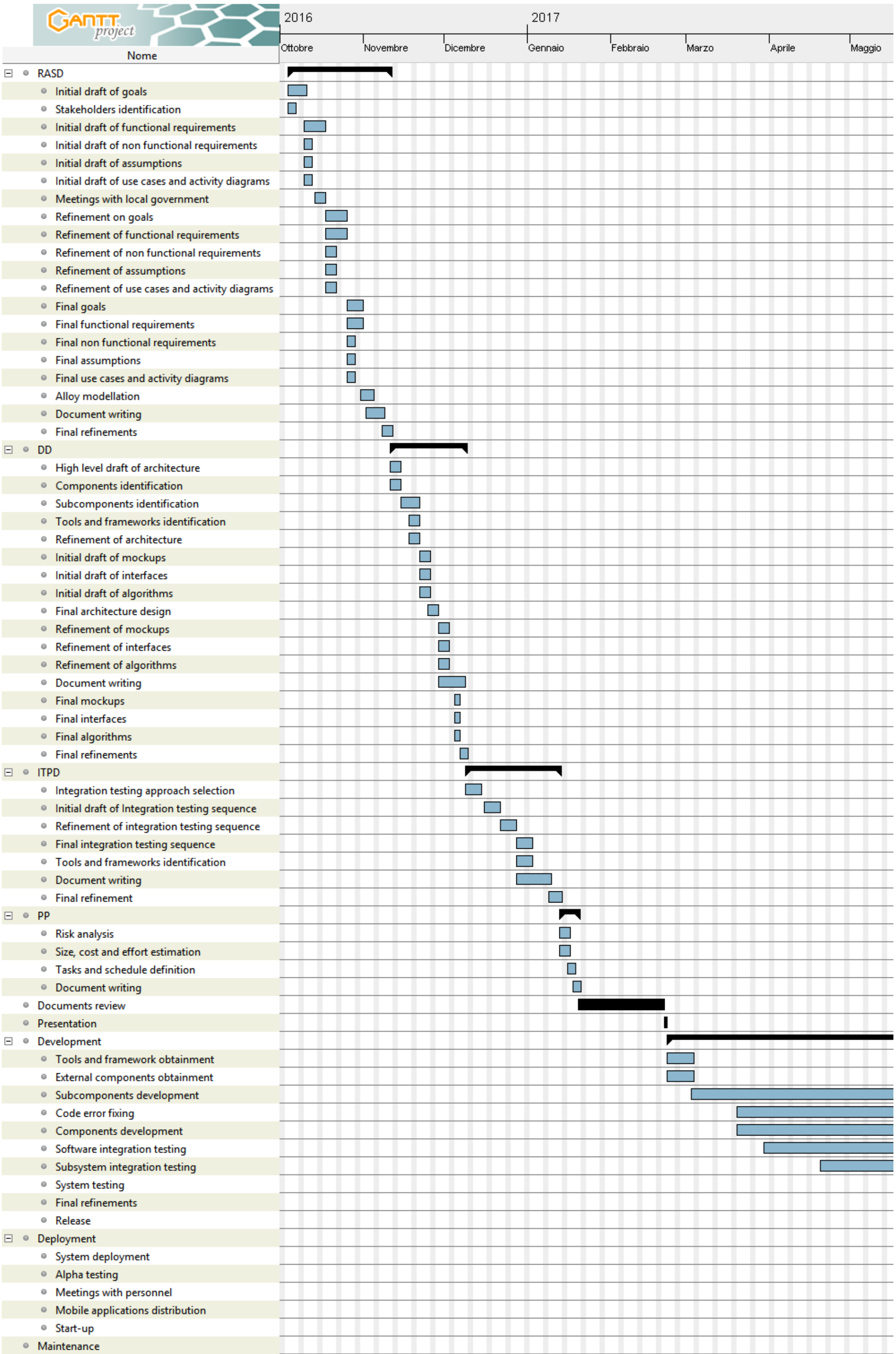
We would need 4 people in order to complete the project in about 13 months, but since our team consists of 3 people, a reasonable development time would be about **17** months. This last value is obviously calculated turning around the last formula.

3 Tasks and schedule

In this section we are going to provide which are the tasks and the subtasks required by the project to be completed. For the documentation, which includes RASD, DD, ITPD and PP, we will consider the deadlines given by the Software Engneering 2 course and, for all that concern the development and the post-development, we will consider the immediately following period.

The final schedule is now summarized and completely described in the following figures.

- RASD [04/10/2016 - 13/11/2016]
- DD [14/11/2016 - 11/12/2016]
- ITPD [12/12/2016 - 15/01/2017]
- PP [16/01/2017 - 22/01/2017]
- Documents review [23/01/2017 - 21/02/2017]
- Presentation [22/01/2017]
- Development [23/01/2017 - 25/08/2017]
- Deployment [26/08/2017 - 10/10/2017]
- Maintenance [11/10/2017 - undetermined]

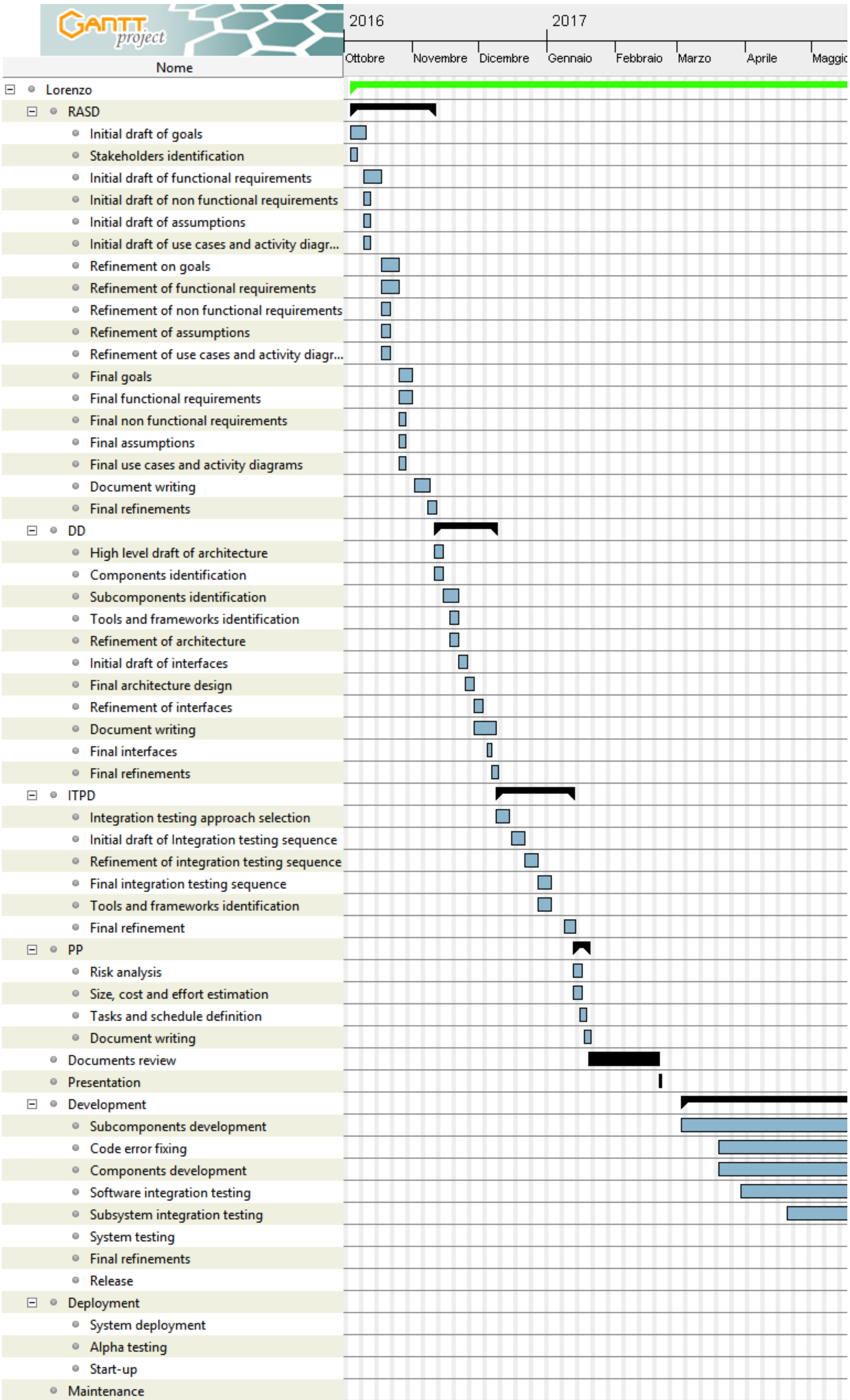

GANTT project
Nome
2016
2017
Ottobre
Novembre
Dicembre
Gennaio
Febbraio
Marzo
Aprile
Maggio
RASD
Initial draft of goals
Stakeholders identification
Initial draft of functional requirements
Initial draft of non functional requirements
Initial draft of assumptions
Initial draft of use cases and activity diagrams
Meetings with local government
Refinement on goals
Refinement of functional requirements
Refinement of non functional requirements
Refinement of assumptions
Refinement of use cases and activity diagrams
Final goals
Final functional requirements
Final non functional requirements
Final assumptions
Final use cases and activity diagrams
Alloy modellation
Document writing
Final refinements
DD
High level draft of architecture
Components identification
Subcomponents identification
Tools and frameworks identification
Refinement of architecture
Initial draft of mockups
Initial draft of interfaces
Initial draft of algorithms
Final architecture design
Refinement of mockups
Refinement of interfaces
Refinement of algorithms
Document writing
Final mockups
Final interfaces
Final algorithms
Final refinements
ITPD
Integration testing approach selection
Initial draft of Integration testing sequence
Refinement of integration testing sequence
Final integration testing sequence
Tools and frameworks identification
Document writing
Final refinement
PP
Risk analysis
Size, cost and effort estimation
Tasks and schedule definition
Document writing
Documents review
Presentation
Development
Tools and framework obtainment
External components obtainment
Subcomponents development
Code error fixing
Components development
Software integration testing
Subsystem integration testing
System testing
Final refinements
Release
Deployment
System deployment
Alpha testing
Meetings with personnel
Mobile applications distribution
Start-up
Maintenance

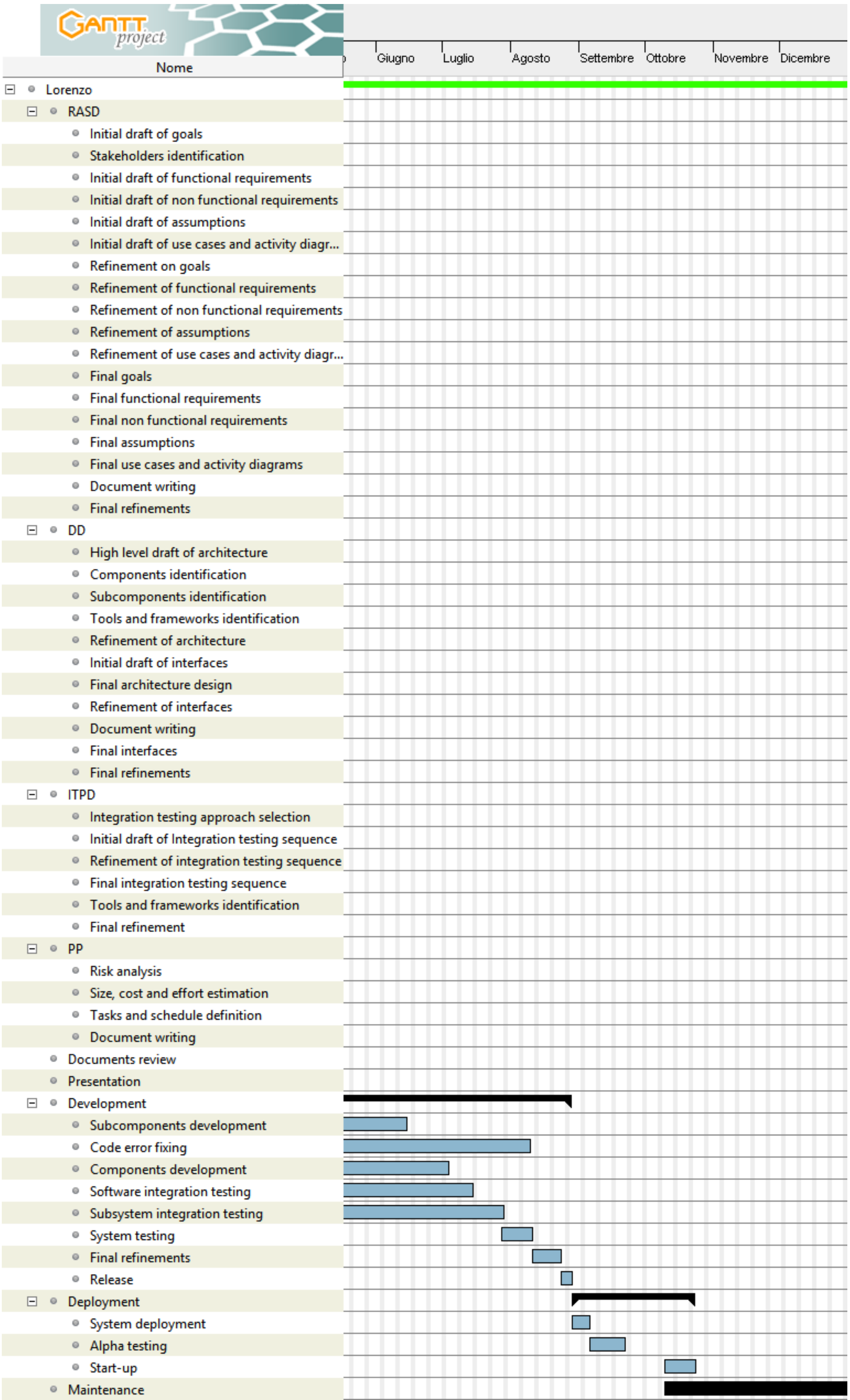

GANTT project
Nome
Giugno
Luglio
Agosto
Settembre
Ottobre
Novembre
Dicembre
RASD
Initial draft of goals
Stakeholders identification
Initial draft of functional requirements
Initial draft of non functional requirements
Initial draft of assumptions
Initial draft of use cases and activity diagrams
Meetings with local government
Refinement on goals
Refinement of functional requirements
Refinement of non functional requirements
Refinement of assumptions
Refinement of use cases and activity diagrams
Final goals
Final functional requirements
Final non functional requirements
Final assumptions
Final use cases and activity diagrams
Alloy modellation
Document writing
Final refinements
DD
High level draft of architecture
Components identification
Subcomponents identification
Tools and frameworks identification
Refinement of architecture
Initial draft of mockups
Initial draft of interfaces
Initial draft of algorithms
Final architecture design
Refinement of mockups
Refinement of interfaces
Refinement of algorithms
Document writing
Final mockups
Final interfaces
Final algorithms
Final refinements
ITPD
Integration testing approach selection
Initial draft of Integration testing sequence
Refinement of integration testing sequence
Final integration testing sequence
Tools and frameworks identification
Document writing
Final refinement
PP
Risk analysis
Size, cost and effort estimation
Tasks and schedule definition
Document writing
Documents review
Presentation
Development
Tools and framework obtainment
External components obtainment
Subcomponents development
Code error fixing
Components development
Software integration testing
Subsystem integration testing
System testing
Final refinements
Release
Deployment
System deployment
Alpha testing
Meetings with personnel
Mobile applications distribution
Start-up
Maintenance

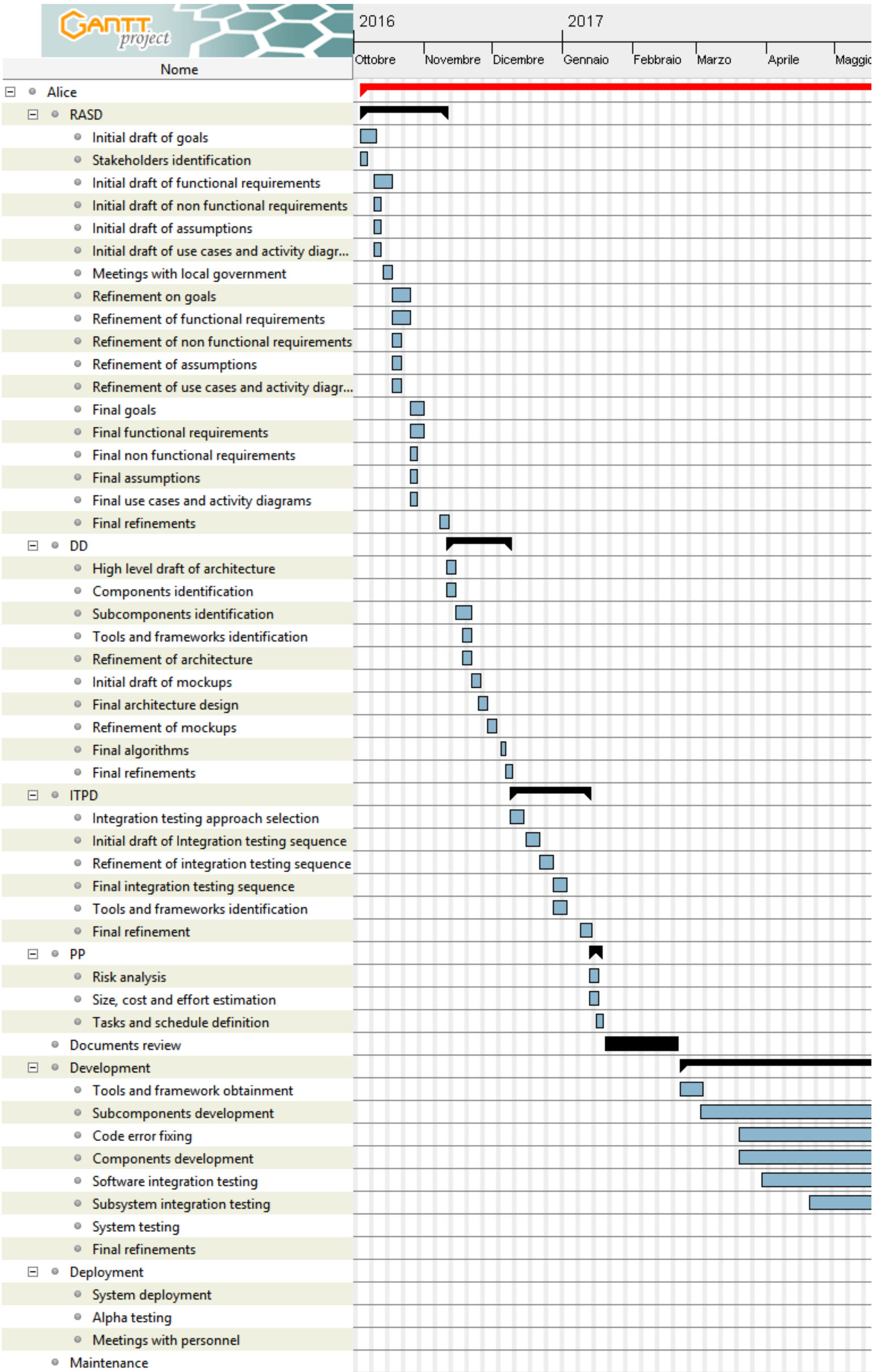
4 Resource allocation

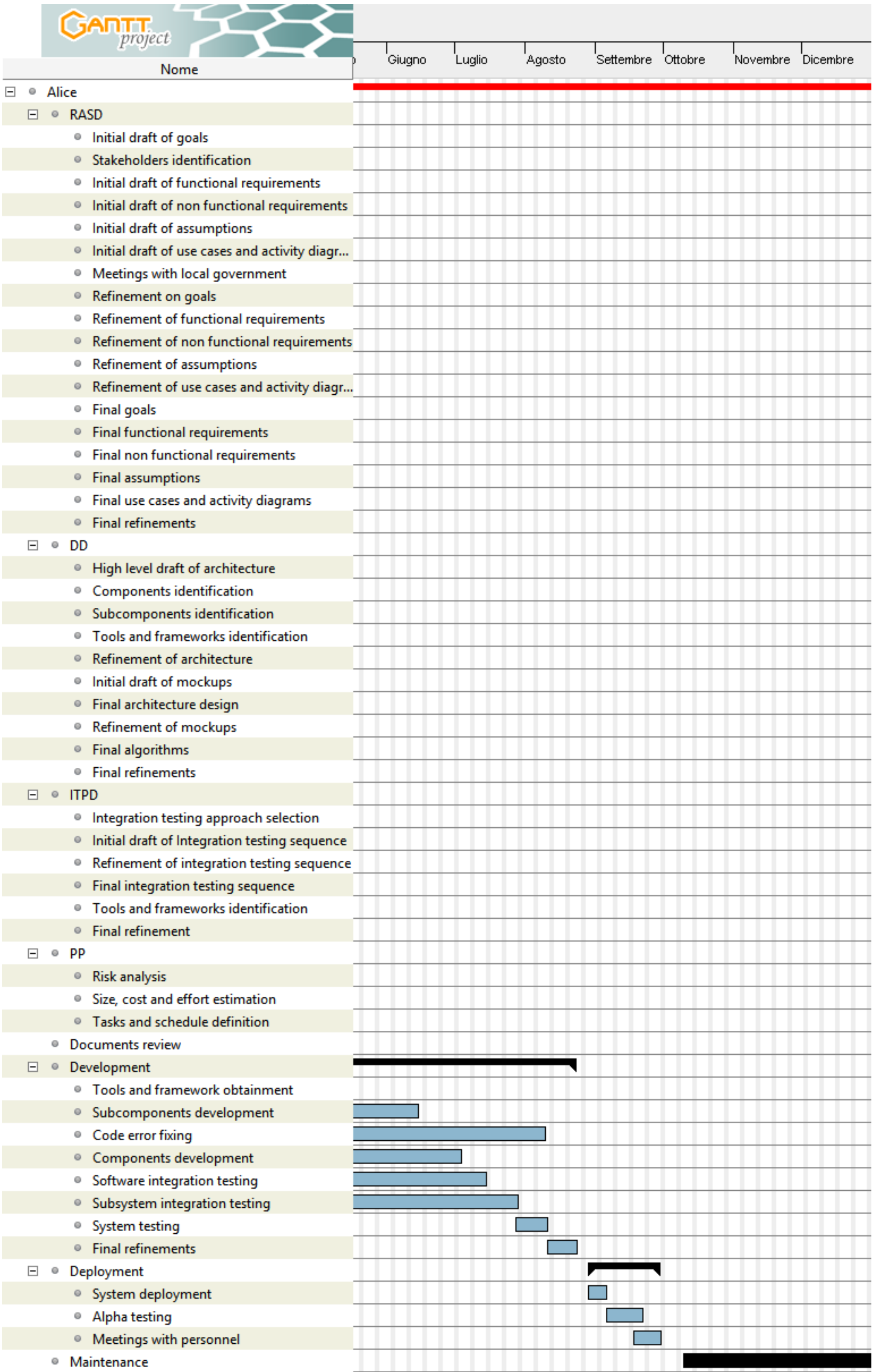
In this section we are going to provide a possible resources allocation for the schedule defined in the previous section. Tasks and subtasks will be divided between the three member of our team, i.e., our resources, which have been introduced in Section 2.2.

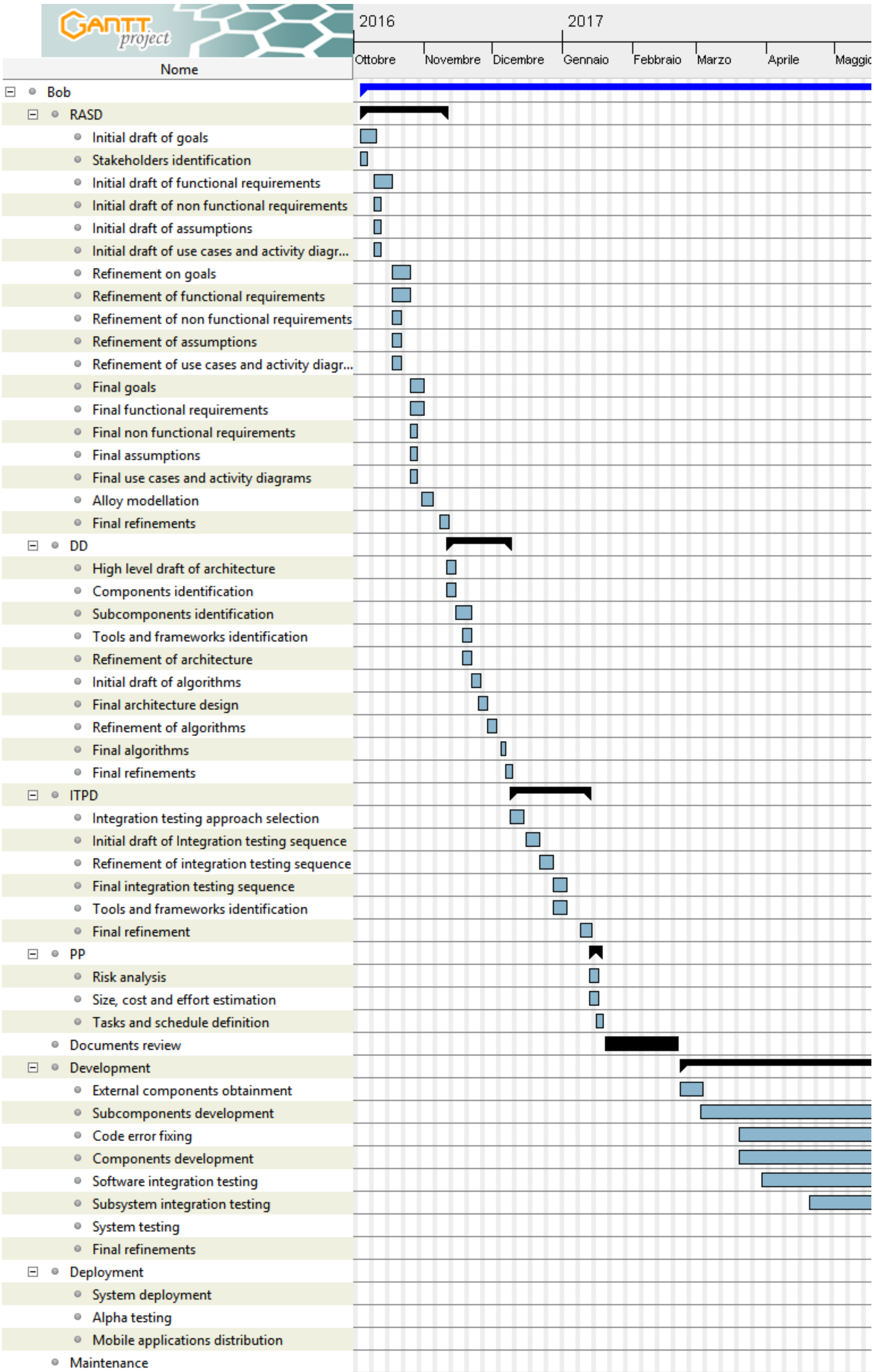
It's important to note that a subtask could be performed by all the members of the team because, the decisions made and the knowledge obtained at the completion of the subtask, could be essential in order to perform other subtasks.


GANTT project
Nome
2016
2017
Ottobre
Novembre
Dicembre
Gennaio
Febbraio
Marzo
Aprile
Lorenzo
RASD
Initial draft of goals
Stakeholders identification
Initial draft of functional requirements
Initial draft of non functional requirements
Initial draft of assumptions
Initial draft of use cases and activity diagr...
Refinement on goals
Refinement of functional requirements
Refinement of non functional requirements
Refinement of assumptions
Refinement of use cases and activity diagr...
Final goals
Final functional requirements
Final non functional requirements
Final assumptions
Final use cases and activity diagrams
Document writing
Final refinements
DD
High level draft of architecture
Components identification
Subcomponents identification
Tools and frameworks identification
Refinement of architecture
Initial draft of interfaces
Final architecture design
Refinement of interfaces
Document writing
Final interfaces
Final refinements
ITPD
Integration testing approach selection
Initial draft of Integration testing sequence
Refinement of integration testing sequence
Final integration testing sequence
Tools and frameworks identification
Final refinement
PP
Risk analysis
Size, cost and effort estimation
Tasks and schedule definition
Document writing
Documents review
Presentation
Development
Subcomponents development
Code error fixing
Components development
Software integration testing
Subsystem integration testing
System testing
Final refinements
Release
Deployment
System deployment
Alpha testing
Start-up
Maintenance

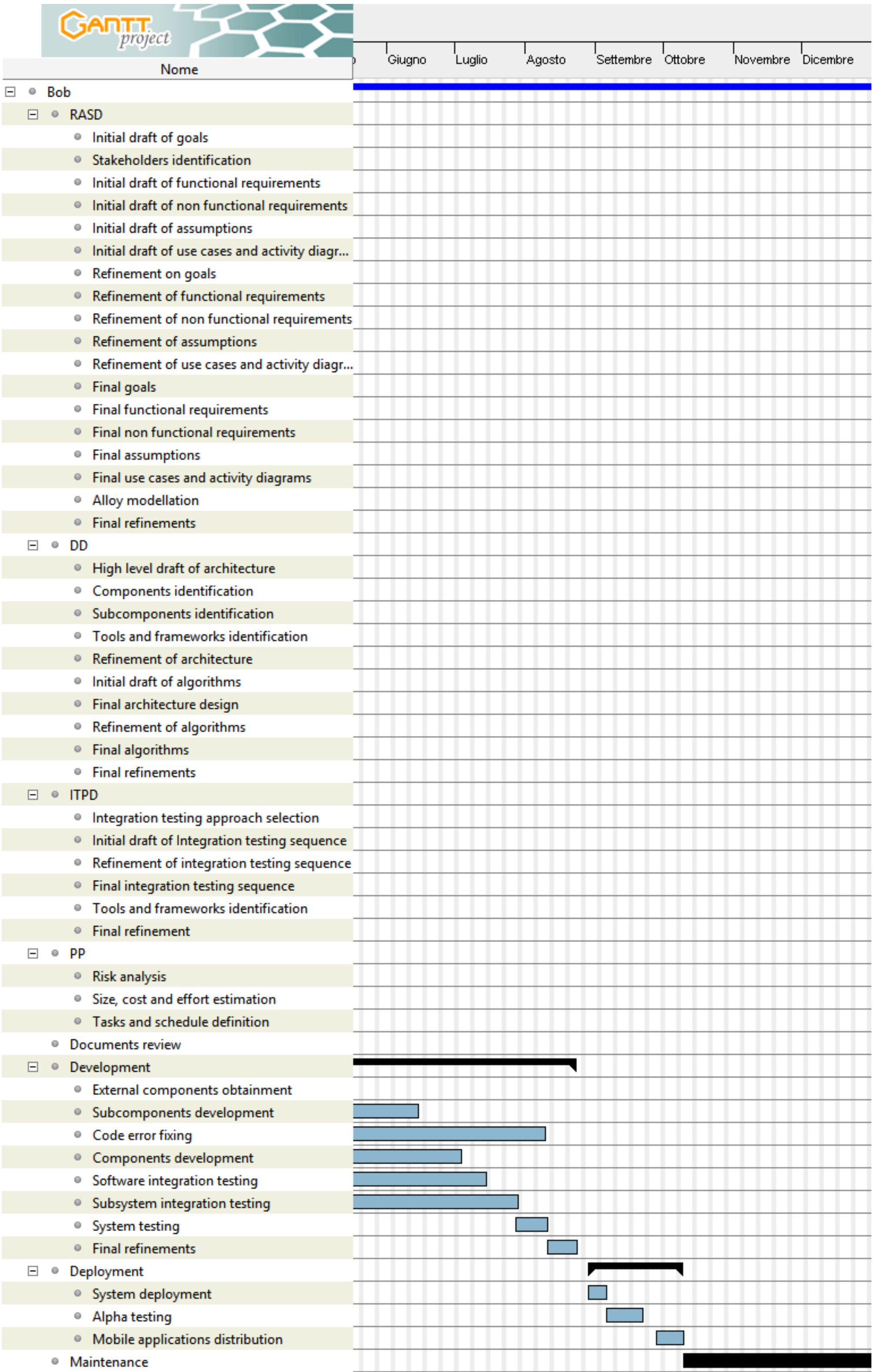

GANTT project
Nome
Giugno
Luglio
Agosto
Settembre
Ottobre
Novembre
Dicembre
Lorenzo
RASD
Initial draft of goals
Stakeholders identification
Initial draft of functional requirements
Initial draft of non functional requirements
Initial draft of assumptions
Initial draft of use cases and activity diagr...
Refinement on goals
Refinement of functional requirements
Refinement of non functional requirements
Refinement of assumptions
Refinement of use cases and activity diagr...
Final goals
Final functional requirements
Final non functional requirements
Final assumptions
Final use cases and activity diagrams
Document writing
Final refinements
DD
High level draft of architecture
Components identification
Subcomponents identification
Tools and frameworks identification
Refinement of architecture
Initial draft of interfaces
Final architecture design
Refinement of interfaces
Document writing
Final interfaces
Final refinements
ITPD
Integration testing approach selection
Initial draft of Integration testing sequence
Refinement of integration testing sequence
Final integration testing sequence
Tools and frameworks identification
Final refinement
PP
Risk analysis
Size, cost and effort estimation
Tasks and schedule definition
Document writing
Documents review
Presentation
Development
Subcomponents development
Code error fixing
Components development
Software integration testing
Subsystem integration testing
System testing
Final refinements
Release
Deployment
System deployment
Alpha testing
Start-up
Maintenance


GANTT project
Nome
2016
2017
Ottobre
Novembre
Dicembre
Gennaio
Febbraio
Marzo
Aprile
Alice
RASD
Initial draft of goals
Stakeholders identification
Initial draft of functional requirements
Initial draft of non functional requirements
Initial draft of assumptions
Initial draft of use cases and activity diagr...
Meetings with local government
Refinement on goals
Refinement of functional requirements
Refinement of non functional requirements
Refinement of assumptions
Refinement of use cases and activity diagr...
Final goals
Final functional requirements
Final non functional requirements
Final assumptions
Final use cases and activity diagrams
Final refinements
DD
High level draft of architecture
Components identification
Subcomponents identification
Tools and frameworks identification
Refinement of architecture
Initial draft of mockups
Final architecture design
Refinement of mockups
Final algorithms
Final refinements
ITPD
Integration testing approach selection
Initial draft of Integration testing sequence
Refinement of integration testing sequence
Final integration testing sequence
Tools and frameworks identification
Final refinement
PP
Risk analysis
Size, cost and effort estimation
Tasks and schedule definition
Documents review
Development
Tools and framework obtainment
Subcomponents development
Code error fixing
Components development
Software integration testing
Subsystem integration testing
System testing
Final refinements
Deployment
System deployment
Alpha testing
Meetings with personnel
Maintenance


GANTT project
Nome
Giugno
Luglio
Agosto
Settembre
Ottobre
Novembre
Dicembre
Alice
RASD
Initial draft of goals
Stakeholders identification
Initial draft of functional requirements
Initial draft of non functional requirements
Initial draft of assumptions
Initial draft of use cases and activity diagr...
Meetings with local government
Refinement on goals
Refinement of functional requirements
Refinement of non functional requirements
Refinement of assumptions
Refinement of use cases and activity diagr...
Final goals
Final functional requirements
Final non functional requirements
Final assumptions
Final use cases and activity diagrams
Final refinements
DD
High level draft of architecture
Components identification
Subcomponents identification
Tools and frameworks identification
Refinement of architecture
Initial draft of mockups
Final architecture design
Refinement of mockups
Final algorithms
Final refinements
ITPD
Integration testing approach selection
Initial draft of Integration testing sequence
Refinement of integration testing sequence
Final integration testing sequence
Tools and frameworks identification
Final refinement
PP
Risk analysis
Size, cost and effort estimation
Tasks and schedule definition
Documents review
Development
Tools and framework obtainment
Subcomponents development
Code error fixing
Components development
Software integration testing
Subsystem integration testing
System testing
Final refinements
Deployment
System deployment
Alpha testing
Meetings with personnel
Maintenance



GANTT project
Nome
2016
2017
Ottobre
Novembre
Dicembre
Gennaio
Febbraio
Marzo
Aprile
Maggio
Bob
RASD
Initial draft of goals
Stakeholders identification
Initial draft of functional requirements
Initial draft of non functional requirements
Initial draft of assumptions
Initial draft of use cases and activity diagr...
Refinement on goals
Refinement of functional requirements
Refinement of non functional requirements
Refinement of assumptions
Refinement of use cases and activity diagr...
Final goals
Final functional requirements
Final non functional requirements
Final assumptions
Final use cases and activity diagrams
Alloy modellation
Final refinements
DD
High level draft of architecture
Components identification
Subcomponents identification
Tools and frameworks identification
Refinement of architecture
Initial draft of algorithms
Final architecture design
Refinement of algorithms
Final algorithms
Final refinements
ITPD
Integration testing approach selection
Initial draft of Integration testing sequence
Refinement of integration testing sequence
Final integration testing sequence
Tools and frameworks identification
Final refinement
PP
Risk analysis
Size, cost and effort estimation
Tasks and schedule definition
Documents review
Development
External components obtainment
Subcomponents development
Code error fixing
Components development
Software integration testing
Subsystem integration testing
System testing
Final refinements
Deployment
System deployment
Alpha testing
Mobile applications distribution
Maintenance


GANTT project
Nome
Giugno
Luglio
Agosto
Settembre
Ottobre
Novembre
Dicembre
Bob
RASD
Initial draft of goals
Stakeholders identification
Initial draft of functional requirements
Initial draft of non functional requirements
Initial draft of assumptions
Initial draft of use cases and activity diagr...
Refinement on goals
Refinement of functional requirements
Refinement of non functional requirements
Refinement of assumptions
Refinement of use cases and activity diagr...
Final goals
Final functional requirements
Final non functional requirements
Final assumptions
Final use cases and activity diagrams
Alloy modellation
Final refinements
DD
High level draft of architecture
Components identification
Subcomponents identification
Tools and frameworks identification
Refinement of architecture
Initial draft of algorithms
Final architecture design
Refinement of algorithms
Final algorithms
Final refinements
ITPD
Integration testing approach selection
Initial draft of Integration testing sequence
Refinement of integration testing sequence
Final integration testing sequence
Tools and frameworks identification
Final refinement
PP
Risk analysis
Size, cost and effort estimation
Tasks and schedule definition
Documents review
Development
External components obtainment
Subcomponents development
Code error fixing
Components development
Software integration testing
Subsystem integration testing
System testing
Final refinements
Deployment
System deployment
Alpha testing
Mobile applications distribution
Maintenance

5 Risk management

The PowerEnJoy project could face several risks during and after its development, thus it's important to consider them in order to avoid any resources wasting. These risks can be divided into 3 main groups: project risks, technical risks and economical risks.

5.1 Project risks

Lack of experience: the team is developing its first project. This will probably slow down the development of the project.

Misunderstanding of the assignment: the final product could be different from the one intended by the customer (in our case, by the professor). In order to avoid this risk, it's important to keep a good communication between the two parties, clarifying all those aspects not completely clear.

Delays over the expected deadlines: for some reasons, the project could require more time than expected. If that happens, we should provide, within the final deadlines, a release which contains at least the core functionalites. All the missing functionalities will be developed later on.

Requirements incompleteness: requirements could be not completely defined. This could generate issues or errors later on, forcing the team to review aspects and slowing down the development of the project.

5.2 Technical risks

Components infeasibility: components could be infeasible. In our case, the component which we should worry about is the car. It is equipped with a computer and many devices, thus it couldn't be possible to provide enough charge to the entire car system. If that happens, we should find another way to manage the car and considering that we are close to the deployment phase, it could cost a lot of resources.

Scalability issues: the system has been thought to be scalable, but we could not be able to obtain the necessary hardware because of economical reasons. We can understand if there's the possibility to face this risk through a good statistical analysis.

External components issues: our system performs some tasks through external services (payments handling, sms dispatching). Changes in term and conditions on these services could pose serious financial or technical problems. In the worst case, we should change service and interface the new one with our system.

Data loss: data can be lost because of hardware failures or misconfigured software. This risk can be prevented implementing an automated backup system.

5.3 Economical risks

Technology innovation: if we would have strong competitors in the current market, we could be forced to adapt the new techologies to our system in order to preserve our state in it.

Regulation change: local and state regulators could change at any time. In the worst case, our service could become infeasible according to the procedures and time limits laid down by law. This risk can be partially avoided by a good feasibility study.

Bankruptcy: the costs for the development, deployment and maintenance could be higher than expected. If they would be also higher than the income from the sales of the software, we could end in bankrupt. This risk can be avoided by a good project estimation.

Insurance laws: changes in insurance laws could make the insurance premiums cost-prohibitive for our system to continue covering them. This risk can't be prevented at all.

Tax laws: changes in tax laws could make our service no longer attractive for consumers and businesses to choose car-sharing over car leasing or car ownership. This risk can't be prevented at all.

A Appendix

A.1 Hours of work

This is the time spent redacting the PP

- Lorenzo Binosi - 30 hours

References

[1] Software Engineering 2 Project, AA 2016/2017 - *Goal and approach, schedule and rules*

[2] Software Engineering 2 Project, AA 2016/2017 - *Assignments 4*

[3] Lorenzo Binosi, *Software Engineering 2: PowerEnJoy - Requirements Analysis and Specification Document*

[4] Lorenzo Binosi, *Software Engineering 2: PowerEnJoy - Design Document.*

[5] Lorenzo Binosi, *Software Engineering 2: PowerEnJoy - Integration Testing Plan Document.*

[6] COCOMO II - Model Definition Manual, Version 2.1, 1995 2000, Center for Software Engineering, USC.
http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf

[7] IFPUG 1994 - *Function Point Counting Practices Manual*, Release 4.0, 1994.