PowerEnJoy

PROJECT PLAN

Flavio Primo, Hootan Haji Manoochehri
POLITECNICO DI MILANO | SOFTWARE ENGINEERING 2

Index

1 Function Points: size estimation 4

1.1 Overview 4

1.2 Internal Logic Files 5

1.3 External Interface File 5

1.4 External Input 5

1.5 External Inquiry 5

1.6 External Output 6

1.7 Results 6

2 COCOMO: effort and cost estimation 6

2.1 Overview 6

2.2 Scale Drivers 6

2.2.1 PREC Precedentedness 7

2.2.2 FLEX Development flexibility 7

2.2.3 RESL Risk resolution 7

2.2.4 TEAM Team cohesion 7

2.2.5 PMAT Process maturity 7

2.3 Cost Drivers 7

2.3.1 RELY Required Software Reliability 7

2.3.2 DATA Database Size 8

2.3.3 CPLX Product Complexity 8

2.3.4 RUSE Required Reusability 8

2.3.5 DOCU Documentation match to lifecycle needs 8

2.3.6 TIME Execution Time Constraint 8

2.3.7 STOR Main Storage Constraint 9

2.3.8 PVOL Platform Volatility 9

2.3.9 ACAP Analyst Capability 9

2.3.10 PCAP Programmer Capability 9

2.3.11 APEX Application Experience 9

2.3.12 PLEX Platform Experience 10

2.3.13 LTEX Language and Tool Experience 10

2.3.14 PCON Personnel continuity 10

2.3.15 TOOL Use of software tools 10

2.3.16 SITE Multisite development 11

2.3.17 SCED Required development schedule 11

2.4 Effort Equation 11

2.5 Schedule Estimation 12
3 Tasks and Schedule 12
4 Resource Allocation 14
5 Risks 16
5.1 Project Risks 16
5.2 Technical Risks 17
5.3 Business Risks 17
6 References 18
7 Hours Spent 18

1 Function Points: size estimation

1.1 Overview

The Function Point approach it's estimation effort tool used in project code and design. Several aspects are considered for the estimation, as described by the specifications:

- **Internal Logic Files:** homogeneous set of data handled by the application being developed;
- **External Interface Files:** homogeneous set of data managed by the application but created elsewhere;
- **External Input:** operation invoked for doing a simple operation on the system with external data (e.g.: user registration, reserving a PowerEnJoy car, …);
- **External Inquiry:** operation that involves both input and output, mainly to retrieve information from the system;
- **External Output:** system operation producing data for the external environment.

For each point a counting weight (Simple, Medium or Complex) has been given according to the parameters specified in Tables 1-3. After that, a certain number of FPs has been calculated for each section per Table 4.

Finally, starting from the total amount of FP, we estimated the project size in SLOC (for more on this, see Section 1.7).

FP Analysis Tables

Record Elements	Data Elements		
	1-19	20-50	51+
1	Simple	Simple	Medium
2-5	Simple	Medium	Complex
6+	Medium	Complex	Complex

Table 1: FP counting weights for "Internal Logic Files" and "External Interface Files"

Record Elements	Data Elements		
	1-5	6-19	20+
0-1	Simple	Simple	Medium
2-3	Simple	Medium	Complex
4+	Medium	Complex	Complex

Table 2: FP counting weights for "External Output" and "External Inquiry"

Record Elements	Data Elements		
	1-4	5-15	16+
0-1	Simple	Simple	Medium
2-3	Simple	Medium	Complex
3+	Medium	Complex	Complex

Table 3: FP counting weights for "External Input"

Function Type	Complexity Weights		
	Simple	Medium	Complex
Internal Logical 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 4: UFP complexity weights

1.2 Internal Logic Files

The application must handle information about the following entities:

File	Record Elements	Data Elements	Counting Weight	FPs
User	6+	51+	Complex	15
PaymentInformation	2-5	51+	Complex	7
Car	6+	51+	Complex	16
Reservation	2-5	51+	Complex	15
Ride	2-5	51+	Complex	15
Address	2-5	51+	Complex	15
SafeArea Special & Normal	2-5	1-19	Simple	7
TOTAL				**89**

Table 5: Internal Logic Files table

1.3 External Interface File

The application must store this information from the external environment:

File	Record Elements	Data Elements	Counting Weight	FPs
Addresses	2-5	51+	Complex	15
Coordinates	2-5	51+	Complex	15
Stripe Payment Info	2-5	51+	Complex	15
TOTAL				**45**

Table 6: External Interface File Table

1.4 External Input

The application must guarantee the following operations for the external environment:

Operation	Entities involved	Data elements	Counting Weight	FPs
registerUser	2	10+	Medium	1x4
loginUser **confirmEmail** **deleteUser**	1	9+	Simple	3x3
getCarByAddress **getCarByRange**	2	11+	Medium	2x4
makeReservation **cancelReservation**	3	16+	Complex	2x6
unlock	4	16+	Complex	1x6
pay	5	16+	Complex	1x6
Total				**45**

Table 7: External Input table

1.5 External Inquiry

The application must make these inquiries available:

operation	Entities involved	Data elements	Counting weight	FPs
getParkingSpots	2	11+	Medium	4
List the billing history of user (through Stripe)	2	15+	Medium	4
Total				**8**

Table 8: External Inquiry table

1.6 External Output

The application produces data to the external environment through the following operations:

operation	Entities involved	Data elements	Counting weight	FPs
Notification of email confirmation	1	9+	Simple	4
Notification of reservation to user	4	20+	Complex	6
Notification of cancel reservation to user	4	20+	Complex	6
Set car status	1	7+	Simple	4
Total				**20**

Table 9: External Output table

1.7 Results

Per [8], the following holds for J2EE:

$$\frac{SLOC}{FP_S} = 46$$

If we sum all the results we got from the previous sections and multiply them by 46, we get:

$$SLOC = 46 \times 207 = 9522$$

2 COCOMO: effort and cost estimation

2.1 Overview

The COCOMO II Cost Estimation Model is a complex estimation technique used by thousands of software engineers all over the world.

It is used to estimate the effort cost of a software engineering project. The core of COCOMO II is the use of the Effort Equation to estimate the number of Person/Month required to develop a complex project.

As a reference, [4] has been used.

2.2 Scale Drivers

Scale Factor	Very low	Low	Nominal	High	Very high	Extra high
PREC SF_i	Thoroughly Unprecedented **6.20**	Thoroughly Unprecedented **4.96**	Thoroughly Unprecedented **3.72**	Generally, familiar **2.48**	Largely familiar **1.24**	Thoroughly familiar **0.00**
FLEX SF_i	Rigorous **5.07**	Occasionally Relaxation **4.05**	Some relaxation **3.04**	Generally, conformity **2.03**	Some conformity **1.01**	Generally, goals **0.00**
RESL SF_i	Little (20%) **7.07**	Some (40%) **5.65**	Often (60%) **4.24**	Generally, (75%) **2.83**	Mostly (90%) **1.41**	Full (100%) **0.00**
TEAM SF_i	Very difficult interaction **5.48**	Some difficulty in interaction **4.38**	Basically cooperative interaction **3.29**	Largely cooperative **2.19**	Highly cooperative **1.10**	Seamless interaction **0.00**
PMAT SF_i	The estimated SW-CMM level 1 lower **7.80**	Equivalent SW-CMM level 1 upper **6.24**	Process maturity SW-CMM level 2 **4.68**	Level (EPML) or SW-CMM level 3 **3.12**	SW-CMM level 4 **1.56**	SW-CMM level 5 **0.00**

Table 10: Scale Factor Values for COCOMO II Models

This section is about COCOMO II Scale Drivers. They are a significant source of exponential variation on a project effort. Each driver has a range of rating levels, from "Very Low" to "Extra High", each with its own rate.

2.2.1 PREC Precedentedness

This driver reflects the previous experience that developers have in this field. This is our first experience, so we think the best value for our team is "Low".

2.2.2 FLEX Development flexibility

This driver will change due to our flexibility degree in the development. Our schedule is quite strict, so we choose "Low" for this project.

2.2.3 RESL Risk resolution

It reflects the extension of risk analysis. A very low value means we have done a little analysis, high means a complete risk analysis. We choose "High" because we did a detailed analysis (Section 5).

2.2.4 TEAM Team cohesion

This value is correlated to how well the development team know each other. In this case, we are a very cooperative team, so "Very high" value is our choice.

2.2.5 PMAT Process maturity

This parameter reflects the process maturity of the organization. In particular, this parameter has been chosen according to a weighted average of "Yes" answers to CMM Maturity Questionnaire. In our case, we have chosen "High" (CMM Level 3).

Scale Driver	Factor	Value
Precedentedness	Low	4,96
Development Flexibility	Low	4,05
Risk Resolution	High	2,83
Team Cohesion	Very High	1,10
Process Maturity	High	3,12
Total		**16,06**

Table 11: Sum of the result

2.3 Cost Drivers

These are the effort multipliers used in COCOMO II model to adjust the nominal effort.

2.3.1 RELY Required Software Reliability

This is the measure of software reliability. "Nominal" is our choice for this case because a downtime would not lead to high financial losses but will cause problems to customers.

RELY descriptors	Slight inconvenience	Easily recoverable losses	Easily recoverable losses	High financial losses	Risk to human life	
Rating Levels	Very low	Low	Nominal	High	Very High	Extra high
Effort Multipliers	0,82	0,92	1,00	1,10	1,26	-

Table 12: RELY descriptors

2.3.2 DATA Database Size

This values tries to estimate effects that large databases could have in our application. We do not have a test database, so we use “Nominal” as value.

DATA descriptors		$\frac{D}{P} < 10$	$10 \leq \frac{D}{P} < 100$	$100 \leq \frac{D}{P} \leq 1000$	$\frac{D}{P} \geq 1000$	
Rating Levels	Very low	Low	Nominal	High	Very High	Extra high
Effort Multipliers	-	0,90	1,00	1,14	1,28	-

Table 13: DATA descriptors

2.3.3 CPLX Product Complexity

According to [8] and [Table 20], our software could be marked as “Nominal”.

Rating Levels	Very low	Low	Nominal	High	Very High	Extra high
Effort Multipliers	0,73	0,87	1,00	1,17	1,34	1,74

Table 14: CPLX descriptors

2.3.4 RUSE Required Reusability

Reusability is useful. Some parts should be designed as reusable (e.g. Mobile communication drivers). Those parts could be used not only in this project. “High” is our choice here.

RUSE descriptors		None	Across project	Across program	Across product line	Across multiple product line
Rating Levels	Very low	Low	Nominal	High	Very High	Extra high
Effort Multipliers	-	0,95	1,00	1,07	1,15	1,24

Table 15: RUSE descriptors

2.3.5 DOCU Documentation match to lifecycle needs

This is a cost driver for the level of required documentation. In our case, it is suitable as “Nominal”.

DOCU descriptors	Many lifecycle needs uncovered	Some lifecycle needs uncovered	Rightsized to lifecycle needs	Excessive for lifecycle needs	Very excessive for lifecycle needs	
Rating Levels	Very low	Low	Nominal	High	Very High	Extra high
Effort Multipliers	0,81	0,91	1,00	1,11	1,23	-

Table 16: DOCU descriptors

2.3.6 TIME Execution Time Constraint

This is a measure of the execution time constraint. We don’t have strict constraints in this case, so we will set it as “Low”.

Rating Levels	Very low	Low	Nominal	High	Very High	Extra high
Effort Multipliers	-	-	1,00	1,11	1,29	1,63

Table 17: TIME descriptors

2.3.7 STOR Main Storage Constraint

This is a measure of the degree of main storage constraint. We don't have any constraint, so we will set it as "Low".

Rating Levels	Very low	Low	Nominal	High	Very High	Extra high
Effort Multipliers	-	-	1,00	1,05	1,17	1,46

Table 18: STOR descriptors

2.3.8 PVOL Platform Volatility

Our estimation is that this is a stable system with low volatility. "Low" is a good choice here.

PVOL descriptors		Major: 12 months Minor: 1 month	Major: 6 months Minor: 2 weeks	Major: 2 months Minor: 1 week	Major: 2 weeks Minor: 2 days	
Rating Levels	Very low	Low	Nominal	High	Very High	Extra high
Effort Multipliers	-	0,87	1,00	1,15	1,30	-

Table 19: PVOL descriptors

2.3.9 ACAP Analyst Capability

This driver should be set to "High" since we dedicated a lot of effort in analyzing the problem requirements.

Rating Levels	Very low	Low	Nominal	High	Very High	Extra high
Effort Multipliers	1,42	1,19	1,00	0,85	0,71	-

Table 20: ACAP Cost Driver

2.3.10 PCAP Programmer Capability

This driver should emphasize our programmers' capabilities as a team. Our cooperation is quite good, so we set it as "High".

Rating Levels	Very low	Low	Nominal	High	Very High	Extra high
Effort Multipliers	1,34	1,15	1,00	0,88	0,76	-

Table 21: PCAP Cost Driver

2.3.11 APEX Application Experience

Our experience in this field is very low. So, we think that a good estimate will happen if we set this value to "Very Low".

APPEX DESCRIPTION	≤ 2 months	6 months	1 year	3 year	6 year	

Rating Levels	Very low	Low	Nominal	High	Very High	Extra high
Effort Multipliers	1,22	1,10	1,00	0,88	0,81	-

Table 22: APEX Descriptors

2.3.12 PLEX Platform Experience

Our average knowledge about platforms as databases, UI, client/server architecture is around 1 year. We set this value as "Nominal".

PLEX DESCRIPTION	≤ 2 months	6 months	1 year	3 year	6 year	
Rating Levels	Very low	Low	Nominal	High	Very High	Extra high
Effort Multipliers	1,19	1,09	1,00	0,91	0,85	-

Table 23: PLEX Descriptors

2.3.13 LTEX Language and Tool Experience

This is like the previous parameter. Our experience is around one year, so this value will be set to "Nominal".

LTEX DESCRIPTION	≤ 2 months	6 months	1 year	3 year	6 year	
Rating Levels	Very low	Low	Nominal	High	Very High	Extra high
Effort Multipliers	1,20	1,09	1,00	0,91	0,84	-

Table 24: LTEX Descriptors

2.3.14 PCON Personnel continuity

We can estimate a "High" personnel continuity.

PCON DESCRIPTION	48% per Year	24% per Year	12% per Year	6% per Year	3% per Year	
Rating Levels	Very low	Low	Nominal	High	Very High	Extra high
Effort Multipliers	1,29	1,12	1,00	0,90	0,81	-

Table 25: PCON Descriptors

2.3.15 TOOL Use of software tools

We are going to use basic tools like Eclipse as IDE, Maven as dependency manager and GIT as versioning tool. So, we think that "Nominal" will be good for us.

Tool DESCRIPTION	Edit, code, debug	Simple, frontend, backend CASE, little integration	Basic lifecycle tools, moderately integrated	Strong, mature lifecycle tools, moderately integrated	Strong, mature, proactive lifecycle tools, well Integrated with processes, methods, reuse	
Rating Levels	Very low	Low	Nominal	High	Very High	Extra high
Effort Multipliers	1,17	1,09	1,00	0,90	0,78	-

Table 22: Tool Descriptors

2.3.16 **SITE** Multisite development

We are going to use emails, interactive maps, push notification services. So, we choose “High” here.

SITE DESCRIPTION	Some phone, mail	Individual phone, FAX	Narrowband email	Wideband electronic communication	Wideband elect. comm, occasional video conf	Interactive multimedia
Rating Levels	Very low	Low	Nominal	High	Very High	Extra high
Effort Multipliers	1,22	1,09	1,00	0,93	0,86	0,80

Table 23: SITE Descriptors

2.3.17 **SCED** Required development schedule

One hundred percent is good for us. So, we will choose “Nominal” here.

SCED DESCRIPTION	75% of Nominal	85% of Nominal	100% of Nominal	130% of Nominal	160% of Nominal	
Rating Levels	Very low	Low	Nominal	High	Very High	Extra high
Effort Multipliers	1,43	1,14	1,00	1,00	1,00	-

Table 24: SCED Descriptors

2.4 Product

Compute the product of all Cost Drivers.

Cost Driver		Factor	Value
RELY	**Requires Software Reliability**	Nominal	1,00
DATA	**Database Size**	Nominal	1,00
CMPLX	**Product Complexity**	Nominal	1,00
RUSE	**Required Reusability**	High	1,07
DOCU	**Documentation match to lifecycle needs**	Nominal	1,00
TIME	**Execution Time Constraint**	Low	-
STOR	**Main Storage Constraint**	Low	-
PVOL	**Platform Volatility**	Low	0,87
ACAP	**Analyst Capability**	High	0,85
PCAP	**Programmer Capability**	High	0,88
APEX	**Application Experience**	Very Low	1,22
PLEX	**Platform Experience**	Nominal	1,00
LTEX	**Language and Tool Experience**	Nominal	1,00
PCON	**Personnel continuity**	High	0,90
TOOL	**Use of software tools**	Nominal	1,00
SITE	**Multisite development**	High	0,93
SCED	**Required development schedule**	Nominal	1,00
PRODUCT			0,71

Table 25: Product of cost driver results

2.5 Effort Equation

Now, having both cost drivers product and scale drivers factors we can compute the effort, in Person-Month with the following equation:

$$Effort = A \times EAF \times KSLOC^{E}$$

Where:

- **A**, the COCOMO 2000 constant, $A = 2{,}94$
- **EAF**, product of all cost drivers. In our case, it is $EAF = 0{,}71$.
- **KSLOC**, using function points estimation $KSLOC = 9{,}522$.
- **E**, is the exponent derived from Scale Drivers. Calculated with the following formula:

$$E = B + 0{,}01 \times \sum_{i=1}^{5} SF_i$$

  Where:
  - **B**, in COCOMO 2000 we have $B = 0{,}91$.

  In our project, we can derive that:

$$E = 0{,}91 + 0{,}01 \times (4{,}96 + 4{,}05 + 2{,}83 + 1{,}10 + 3{,}124) = 0{,}91 + 0{,}01 \times 16{,}064$$
$$= 0{,}91 + 0{,}16064 \cong 1{,}07$$

Using these parameters, we can compute our effort:

$$Effort = 2{,}94 \times 0{,}71 \times 9{,}522^{1{,}07} = 23{,}26\ PM$$

2.6 Schedule Estimation

Now we can estimate the project duration with the following equation:

$$Duration = 3.67 * Effort^{F}$$

Where Effort is the estimated effort and SE is the schedule equation exponent derived from the five Scale Drivers. We can obtain SE using the following formula:

$$F = D + 0.2 * 0.01 * \sum_{i=1}^{5} \mathrm{SF}_i = 0.28 + 0.2 * 0.01 * 16.06 \cong 0.31$$

$$Duration = 3.67 * 23.26^{F} = 3.67 * 23.26^{0.31} \cong 9.73\ Month$$

3 Tasks and Schedule

The main tasks of this project are the following:

1. **RASD [1] creation**, which explains in detail functional and nonfunctional requirements, domain assumption and goals of the application to be built.
2. **DD [2] creation**, which deals with the architecture and the design shape of the application.
3. **ITPD [3] creation**, which contains integration testing strategy we intend to apply to the application.
4. **PP creation**, this very document.
5. **Quick presentation creation**, using slides, (roughly 10 min) of the previously mentioned documents to the client.
6. **Application development** and **unit tests preparation**.
7. **Run integration tests** on the application.

For the first tasks the activities were already given along with corresponding deadlines for the submission of needed documents. Starting from the implementation, instead, no schedule was given so, according to the COCOMO estimation performed and described in 2, we expect the

implementation of the application to be complete in ~9.5 months, around the 16th of September 2017. Regarding the integration testing, it will take place in the last month of development.

The development of the application started after the creation of the Design Document and will be carried on in parallel with the rest of the tasks.

Tests will be run on the developing application to verify the proper functioning of every new functionality added.

In Figure 1 you can find the dependency graph of every task, in Table 30, instead, you can find the schedule of every task. Also, the Gantt chart for the project is provided in Figure 2.

We redefined deadlines like this: 13th Nov 2016 at 23:59 is 14th Nov 2016 at 00:00 AM.
The reason behind this modification is that, we used Microsoft Visio for doing our project, and since, the deadlines were on Sunday evening and Visio doesn't accept that date as a working date.

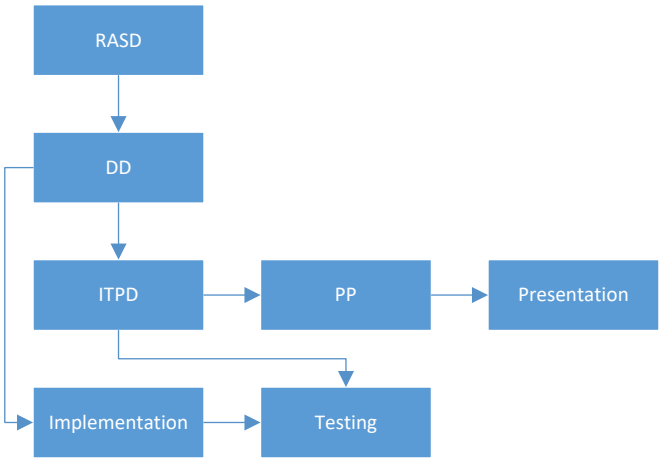

Figure 1: Dependencies between tasks

Activity	Start Date	Deadline
RASD	17/10/2016	14/11/2016
DD	15/11/2016	12/12/2016
ITPD	20/12/2016	02/01/2017
Project Plan	03/01/2017	17/01/2017
Presentation	03/02/2017	20/02/2017
Implementation	20/12/2016	05/09/2017
Integration Testing	06/09/2017	13/10/2017

Table 30: Schedule for project tasks

ID	Task Name	Start	Finish	Duration
1	RASD	10/17/2016	11/14/2016	21d
2	DD	11/15/2016	12/12/2016	20d
3	ITPD	12/20/2016	1/2/2017	10d
4	PROJECT PLAN	1/3/2017	1/17/2017	11d
5	Presentation	2/3/2017	2/20/2017	12d
6	IMPLEMENTATION	12/20/2016	9/5/2017	186d
7	INTEGRATION TESTING	9/6/2017	10/13/2017	28d

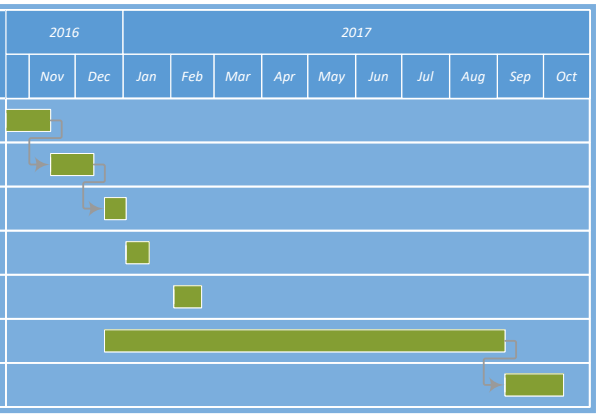

Figure 2: Gantt chart of the project

4 Resource Allocation

This section is meant to show how the available resources are allocated to the project.

Every assigned task is divided in two macro areas and each one of them is assigned to a member of the team. As you can notice, every member works on all the tasks; in this way, the time needed to complete a task is a bit more but we also increase the overall awareness of every member about the project itself, reducing the possibility to create misunderstandings. At the end of every task, the whole team is asked to revision the document before the submission, the week before the deadline.

Regarding the implementation and the integration testing, each member of the team is asked to focus on a tier of the application, as soon as the Design Document is complete, starting in parallel with another task and then focusing on the implementation itself. After one functionality is complete, the testing related will be carried on by another member of the team, to make the unit test more accurate.

Please refer to Table 30, Figure 2 for a better understanding of the division of the work.

The division of work between team members is shown in tables 31, 32, 33, 34, 35, 36, 37.

Resource	**2016-10-17 to 2016-11-14**			
	Week 1	Week 2	Week 3	Week 4
Flavio	- Introduction - Scenarios - Requirements	- Use Cases - Diagrams	- Alloy	- Revision - Presentation
Hootan	- Introduction - Scenarios - Requirements	- Use Cases - Diagrams	- Interfaces	- Revision - Presentation

Table 31: RASD resource allocation

Resource	**2016-11-15 to 2016-12-12**			
	Week 1	Week 2	Week 3	Week 4
Flavio	- Introduction - Runtime View - Deployment View	- Component View - Runtime View	- Algorithm Design - Comp. interfaces	- Revision - Presentation

Hootan	- Introduction - Runtime View - Deployment View	- Component View - Runtime View	- User Interface Design - Component View	- Revision - Presentation

Table 32: DD resource allocation

Resource	**2016-12-20 to 2017-01-02**	
	Week 1	Week 2
Flavio	- Introduction - Test description	- Integration strategy - Stubs and test data - Revision
Hootan	- Introduction - Test description	- Integration strategy - Stubs and test data - Revision

Table 33: ITPD resource allocation

Resource	**2016-01-03 to 2017-01-17**	
	Week 1	Week 2
Flavio	- Function points - Task and schedule - COCOMO II	- Risks - Resource allocation - Revision
Hootan	- Function points - Task and schedule - COCOMO II	- Risks - Resource allocation - Revision

Table 34: Resource allocation for Planning

Resource	**2016-02-03 to 2017-02-20**	
	Week 1	Week 2
Flavio	- Slide	-
Hootan	- Slide	-

Table 35:: Resource allocation for Presentation

Resource	**2016-12-20 to 2017-09-05**			
	1st – 3rd months	4th and 5th months	6th and 7th months	8th and 9th months
Flavio	- Data Base Tier - Business Tier	- Business Tier - integration with powerEnjoy Box	- API Services	- Test DB and business tier (Hootan part) - Test mobile app
Hootan	- Data Base Tier - Business Tier	- Business Tier - integration with powerEnjoy Box	- Mobile application	- Test DB and business tier (Flavio part) - Test APIs

Table 36: Resource allocation for Implementation

Resource	2017-09-06 to 2017-10-13	
	1st and 2nd Weeks	3rd and 4th Weeks
Flavio	- Data - Business Tier	- Business Tier - Mobile App
Hootan	- Data - Business Tier	- Business Tier - APIs

Table 67: Resource allocation for Integration Testing

5 Risks

Risks have to be considered in a complete project planning, owing to their uncertain ad dangerous nature. A sudden change in mind, actions, economical situations and alike could drift the project development into failure; this is the reason why they are here analyzed. Three main risk categories will be later described:

- **Project risks:** involving the project plan (described in these pages).
  Project schedule and overall costs may be subject to (worse) changes due to these risks.
- **Technical risks:** involving the actual implementation of the project.
  They may affect the quality of the software being developed.
- **Business risks:** involving the company developing the software.
  This may cause trouble to the project (e.g.: if the business cannot subsidize the software being developed anymore).

5.1 Project Risks

- **Risk:** No estimations/schedules have been made before this project. A lack of experience in this area can lead to serious errors in evaluating development time.
  - *Probability:* High
  - *Damage:* Critical
  - *How to deal with it:* Study previous works on a similar subject can be very helpful.
- **Risk:** Due to several overlapping tasks the team is involved into, the project is very likely to suffer from schedule delays.
  - *Probability:* High
  - *Damage:* Critical
  - *How to deal with it:* A strict organization among the team components is fundamental. This implies a constant cooperation between developers.
- **Risk**: Collaboration issues can sometimes be crucial, especially when dealing with task divisions.
  - *Probability:* Medium
  - *Damage:* Critical
  - *How to deal with it:* Meeting often can be a solution, other than explicitly writing whose responsibility for each task is.
- **Risk**: The team is very small (only 2 people) but homogeneous; if someone leaves or gets ill then the remaining team would have serious repercussions.
  - *Probability:* Low
  - *Damage:* Catastrophic
  - *How to deal with it:* All team members must be able to cover all development sections and cooperate effectively.

5.2 Technical Risks

- **Risk:** A lack of previous experience in developing with Java EE can surely slow down the entire team, which must study these new technologies first.
    - *Probability:* High
    - *Damage:* Critical
    - *How to deal with it:* This must be accounted in the first stages of planning and inserted in the project scheduling.
- **Risk:** The application may be susceptible to security issues if not well designed.
    - *Probability:* Medium
    - *Damage:* Critical
    - *How to deal with it:* All modern standards in computer security guidelines must be followed, especially when dealing with the user input, which must be correctly verified and processed. OWASP security standard may provide deep insight about this risk [6]
- **Risk**: Testing may prove difficult (for example, if several mocks are necessary) or highlight problems which are hard to solve, especially when doing integration testing or —even worse—validation.
    - *Probability:* Medium
    - *Damage:* Critical
    - *How to deal with it:* All components must be unit tested as soon as possible, to eliminate serious bugs when they first appear; integration testing must be done per the specifications contained in [5]. A periodic check of requirements contained in [2] is also required.
- **Risk**: The application server runs on Amazon Web Services (AWS) as specified in [2]. If Amazon decides to close its cloud division the application server would go down together with the online infrastructure.
    - *Probability:* Low
    - *Damage:* Catastrophic
    - *How to deal with it:* A migration plan to another cloud provider should be prepared and be ready to be applied in such a situation.

5.3 Business Risks

- **Risk:** Testing devices & infrastructure (PCs, several mobile phones, Amazon rent, PowerEnJoy cars, PowerEnJoy boxes) need to be purchased and configured. This is going to increase costs, that may be not sustainable if the company is too small.
    - *Probability:* High
    - *Damage:* Catastrophic
    - *How to deal with it:* Testing tools are to be clearly defined in [3], to avoid worthless spending.
- **Risk:** The company may find itself in serious financial trouble.
    - *Probability:* Low
    - *Damage:* Catastrophic
    - *How to deal with it:* A feasibility study together with the RASD must highlight the impossibility of starting a whole new project.
- **Risk**: PowerEnJoy may violate some laws of metropolitan transportation in the future.
    - *Probability:* Low
    - *Damage:* Critical

    - *How to deal with it:* A periodic check must be done to avoid legal consequences. In the case of drastic changes, the whole team must work to adapt to the new regulations as soon as possible.
- **Risk**: Another company come up with new cars and technologies or with competitive prices.
    - *Probability:* High
    - *Damage:* Critical
    - *How to deal with it:* this risk can be addressed in a complementary way with a business and a technical approach. From the business point of view, it should be put in place a marketing strategy and to make competitive prices. From a technical point of view, it should be provided at least a feature parity with the competitor.

6 References

1. Flavio Primo, Hootan Haji Manoochehri – PowerEnJoy: Requirements Analysis and Specification Document
2. Flavio Primo, Hootan Haji Manoochehri – PowerEnJoy: Design Document
3. Flavio Primo, Hootan Haji Manoochehri – PowerEnJoy: Integration Test Plan Document
4. COCOMO II - Model Definition Manual (http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf)
5. QSM - Function Point Languages Table (http://www.qsm.com/resources/function-point-languages-table)
6. OWASP Top Ten Web Application Vulnerabilities in J2EE (https://www.owasp.org/images/2/2e/OWASP_NL_Top_Ten_Web_Application_Vulnerabilities_in_J2EE.pdf)
7. COCOMO II - Drivers (http://sunset.usc.edu/research/COCOMOII/expert_cocomo/drivers.html)

7 Hours Spent

Table describing the time management for the team.

Team member	**Hours**
Flavio Primo	10
Hootan Haji Manoochehri	10
	20 total