Project Plan

M. Albanese, M. Bianchi, A. Carlucci

February 2, 2016

Contents

Table 2. FP Counting Weights

For Internal Logical Files and External Interface Files

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

For External Output and External Inquiry

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

For External Input

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

(a) FP Counting Weights

Table 3. UFP Complexity Weights

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

(b) UFP Complexity Weights

Figure 1: FP Analysis

1 Function Points: size estimation

1.1 Overview

The **Function Point approach** is a very useful tool for estimating the effort needed in designing and coding a project. Several aspects are considered for the estimation, as prescribed 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 (for example, user registration, booking a cab...);

External Inquiry: operation that involves both input and output, mainly for retrieving information from the system;

External Output: system operation producing data for the external environment.

For each point a **counting weight** (*Low*, *Avg.* or *High*) has been given according to the parameters specified in Figure 1(a). After that, a certain amount of FPs has been calculated for each section according to Figure 1(b).

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

1.2 Internal Logic Files

The application must handle information about the following entities:

File	Record Elements	Data Elements	Counting Weight	FPs
Customer	6+	51+	High	15
Ride	6+	51+	High	15
Call	6+	51+	High	15
TaxiDriver	6+	51+	High	15
Address	2–5	51+	High	15
Zone	2–5	1–19	Low	7
TOT				**82**

1.3 External Interface File

The application must store these information from the external environment:

File	Record Elements	Data Elements	Counting Weight	FPs
Maps	6+	51+	High	10
TOT				**10**

1.4 External Input

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

Operation	Entities involved	Data Elements	Counting Weight	FPs
Login/signup/logout	1	16+	Avg	3×4
Edit/delete profile	1	16+	Avg	2×4
Add/delete request/res	3	16+	High	3×6
Add/remove/update ride	1	16+	Avg	3×4
Set taxi status	1	16+	Avg	4
TOT				**54**

1.5 External Inquiry

The application must make these inquiries available:

Operation	Entities involved	Data Elements	Counting Weight	FPs
Get previous rides	2	20+	High	6
Show calls for a user	3	20+	High	6
Check existence of a ride	1	20+	Avg	4
Get info for a ride	1	20+	Avg	4
Get pre-calculated fee	1	20+	Avg	4
TOT				**24**

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 to drivers (accepting/refusing ride)	2 (Ride & Customer)	20+	High	7
Notification to customers (ride accepted)	2 (Ride & Taxi)	20+	High	7
Confirmation email (after signup)	1	20+	Avg	5
TOT				**19**

1.7 Results

According to [8], the following holds for J2EE:

$$\frac{\text{SLOC}}{\text{FPs}} = 46$$

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

$$\text{SLOC} = 46 \times 189 = 8694$$

2 COCOMO®: effort & 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.

We have used [9] as a reference.

2.2 Scale Drivers

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	The estimated Equivalent Process Maturity Level (EPML) or SW-CMM Level 1 Lower	SW-CMM Level 1 Upper	SW-CMM Level 2	SW-CMM Level 3	SW-CMM Level 4	SW-CMM Level 5
SF_j:	7.80	6.24	4.68	3.12	1.56	0.00

Figure 2: Scale Factor Values for COCOMO®II Models

In this section we will talk 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.

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

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.

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 (Chapter 5).

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.

PMAT Process maturity.
This parameter reflects the process maturity of the organizazion. In particular, this parameter has been choosen 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 1: Sum of the results

2.3 Cost Drivers

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

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 passengers.

RELY Descriptors	Slight inconvenience	Easily recoverable losses	Easily recoverable losses	High financial loss	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	n/a

Table 2: RELY Descriptors

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		D/P < 10	$10 \leq D/P \leq 100$	$100 \leq D/P \leq 1000$	$D/P \geq 1000$	
Rating Levels	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Effort Multipliers	n/a	0.90	1.00	1.14	1.28	n/a

Table 3: DATA Descriptors

CPLEX Product Complexity
According to [10, 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 4: CPLEX Cost Driver

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 lines
Rating Levels	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Effort Multipliers	n/a	0.95	1.00	1.07	1.15	1.24

Table 5: RUSE Descriptors

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	n/a

Table 6: DOCU Descriptors

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	n/a	n/a	1.00	1.11	1.29	1.63

Table 7: TIME Cost Driver

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	n/a	n/a	1.00	1.05	1.17	1.46

Table 8: STOR Cost Driver

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 weeks	Major: 2 weeks. Minor: 2 days	
Rating Levels	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Effort Multipliers	n/a	0.87	1.00	1.15	1.30	n/a

Table 9: PVOL Descriptors

ACAP Analyst Capability
This driver should be set to *High* since we dedicated a lot of effort in analysing 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	n/a

Table 10: ACAP Cost Driver

PCAP Programmer Capability
This driver should emphasize our programmers' capabilites 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	n/a

Table 11: PCAP Cost Driver

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*.

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

Table 12: APEX Descriptors

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 Descriptors	$\leq$ 2 months	6 months	1 year	3 years	6 years	
Rating Levels	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Effort Multipliers	1.19	1.09	1.00	0.91	0.85	n/a

Table 13: PLEX Descriptors

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 Descriptors	$\leq$ 2 months	6 months	1 year	3 years	6 years	
Rating Levels	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Effort Multipliers	1.20	1.09	1.00	0.91	0.84	n/a

Table 14: LTEX Descriptors

PCON Personnel continuity
We can estimate a *High* personnel continuity.

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

Table 15: PCON Descriptors

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 Descriptors	Edit, code, debug	Simple, frontend, backend CASE, little integration	Basic lifecycle tools, moderately integrated	Strong, mature lifecycle tools, moderately integrated	Strong, mature, proactive lifecycle tools, well integrated with processes, methods, reuse	
Rating Levels	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Effort Multipliers	1.17	1.09	1.00	0.90	0.78	n/a

Table 16: TOOL Descriptors

SITE Multisite development
We are going to use chats, emails and phone calls. So we choose *High* here.

SITE Descriptors	Some phone, mail	Individual phone, FAX	Narrowband email	Wideband electronic communi-cation	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 17: SITE Descriptors

SCED Required development schedule
One hundred percent is good for us. So we will choose *Nominal* here.

SCED Descriptors	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	n/a

Table 18: SCED Descriptors

Product Now we can compute the product of all Cost Drivers.

Cost Driver	Factor	Value
Required Software Reliability	`Nominal`	1.00
Database Size	`Nominal`	1.00
Product Complexity	`Nominal`	1.00
Required Reusability	`High`	1.07
Documentation match to lifecycle needs	`Nominal`	1.00
Execution Time Constraint	`Low`	n/a
Main Storage Constraint	`Low`	n/a
Platform Volatility	`Low`	0.87
Analyst Capability	`High`	0.85
Programmer Capability	`High`	0.88
Application Experience	`Very Low`	1.22
Platform Experience	`Nominal`	1.00
Language and Tool Experience	`Nominal`	1.00
Personnel continuity	`High`	0.90
Use of software tools	`Nominal`	1.00
Multisite development	`High`	0.93
Required development schedule	`Nominal`	1.00
Product		0.71

Table 19: Product between Cost driver results

2.4 Effort Equation

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

$$Effort = A * EAF * KSLOC^{E}$$

Where A is the COCOMO®2000 constant, $A = 2.94$. EAF is the product of all cost drivers. In our case it is $EAF = 0.71$. Using function points estimation we can say that $KSLOC = 8.694$. Last thing: E is the exponent derived from Scale Drivers. It is calculated with the following formula:

$$E = B + 0.01 * \sum_{i=1}^{5} SF_i$$

Where $B = 0.91$ in COCOMO®2000. In our project, we can derive that $E = 0.91 + 0.01 * 16.06 = 0.91 + 0.1606 \cong 1.07$.

Using this parameter, we can compute our effort:

$$Effort = 2.94 * 0.71 * 8.694^{1.07} = 21.11\ PM$$

2.5 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:

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

$$Duration = 3.67 * 21.11^{F} = 3.67 * 21.11^{0.31} \cong 9.44\ Months$$

3 Tasks & Schedule

The main tasks of this project are the following:

1. Creation of the **Requirement Analysis and Specification Document** [2], which explains in detail functional and nonfunctional requirements, domain assumption and goals of the application to be built.

2. Creation of the **Design Document** [3], which deals with the architecture and the design shape of the application.

3. Creation of the **Integration Testing Plan Document** [5], which contains integration testing strategy we intend to apply to the application.

4. Creation of the **Project Plan**, this document.

5. Preparation of a quick **presentation**, using slides, ($\sim$ 10 min) of the previously mentioned documents to our client.

6. **Development** of the entire application and the preparation of the unit tests.

7. Running of **integration testing** 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 15th of September 2016. Regarding the integration testing, it will take place in the last month of developement.

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 in order to verify the proper functioning of every new functionality added.

In Figure 3 you can find the dependency graph of every task, in Table 20, instead, you can find the schedule of every task. Also the Gantt chart for the project is provided in Figure 4.

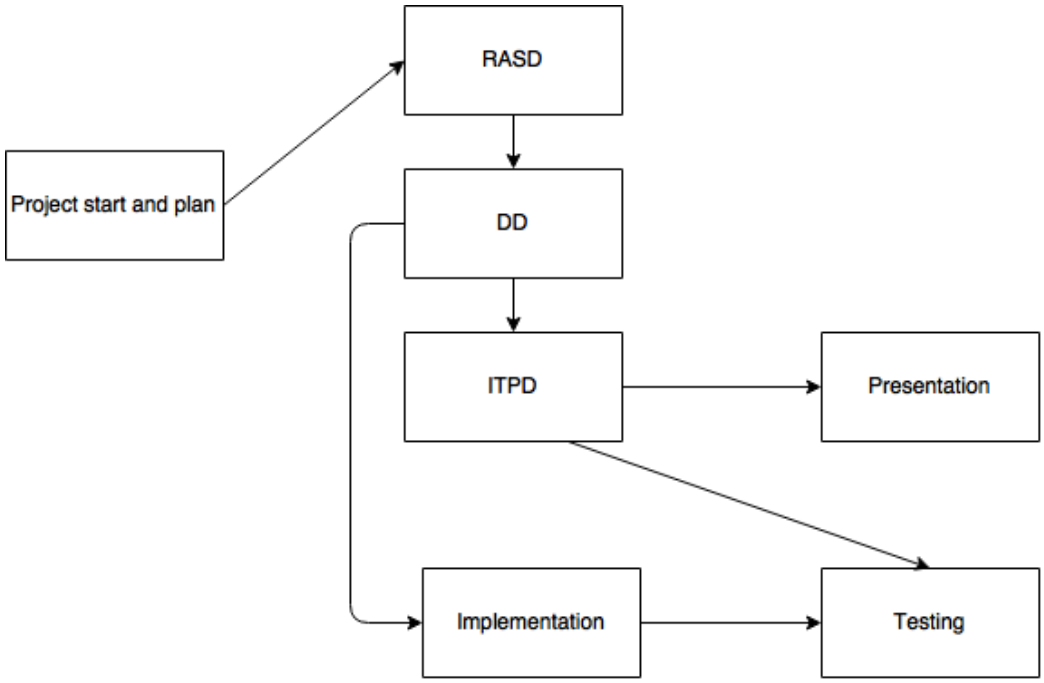

Figure 3: Dependencies between tasks shown as graph.

Activity	**Start Date**	**Deadline**
RASD	2015-10-15	2015-11-06
DD	2015-11-11	2015-12-04
ITPD	2016-01-07	2016-01-21
Project Plan	2016-01-21	2016-02-02
Presentation	2016-02-03	2016-02-12
Implementation	2015-12-05	2016-09-15
Integration Testing	2016-08-01	2016-09-31

Table 20: Schedule for project tasks.

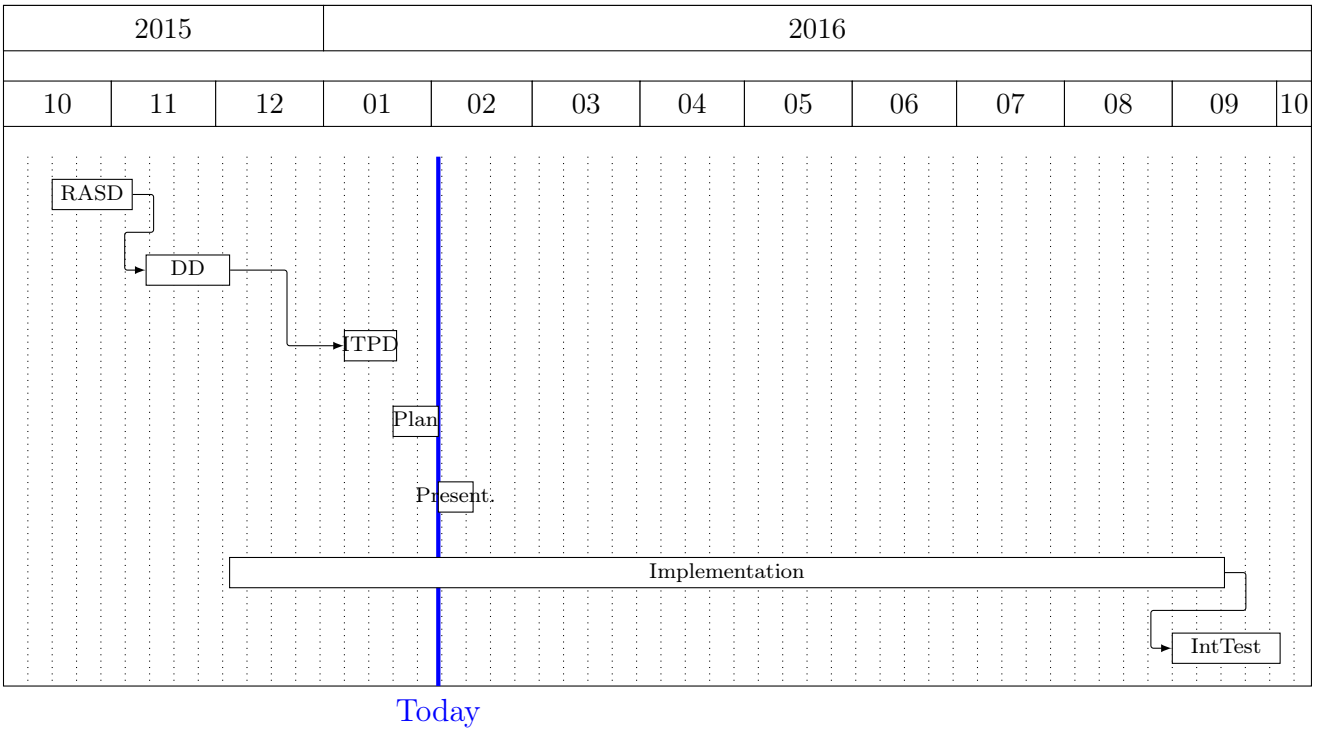

Figure 4: Gantt chart of the project.

4 Resource Allocation

This section is meant to show the way our available resources are allocated to the project. Every assigned task is divided in three macroareas 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 other 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, in order to make the unit test more accurate.

Please refer to Table 20, Figure 4 for a better understanding of the division of the work.

The division of work between team members is shown in tables 21, 22, 23, 24, 25, 26, 27.

Resource	**2015-10-15 to 2015-11-06**			
	1^{st} week	2^{nd} week	3^{rd} week	4^{th} week
Michele	Introduction	Alloy	Alloy	Revision
Mattia	Scenarios	Use Cases	Diagrams	Revision
Alain	Requirements	Requirements	Interfaces	Revision

Table 21: Resource allocation for RASD.

Resource	**2015-11-11 to 2015-12-04**			
	1^{st} week	2^{nd} week	3^{rd} week	4^{th} week
Michele	Introduction	Component view	Component View	Revision
Mattia	Runtime view	Runtime view	Comp. interfaces	Revision
Alain	Deployment view	Algorithm Design	Algorithm Design	Revision

Table 22: Resource allocation for DD.

Resource	**2016-01-07 to 2016-01-21**	
	1^{st} week	2^{nd} week
Michele	Introduction	Integration strategy
Mattia	Test description	Stubs and test data
Alain	Test description	Revision

Table 23: Resource allocation for ITPD.

Resource	**2016-01-21 to 2015-02-02**	
	1^{st} week	2^{nd} week
Michele	Function Points	Risks
Mattia	Task and schedule	Resource allocation
Alain	Cocomo II	Revision

Table 24: Resource allocation for Planning.

Resource	**2016-02-03 to 2016-02-12**	
	1^{st} week	2^{nd} week
Michele		Slide
Mattia		Slide
Alain		Slide

Table 25: Resource allocation for Presentation.

Resource	**2015-12-05 to 2016-09-15**			
	1^{st} trimester	$4^{th}/5^{th}$ month	$6^{rd}/7^{th}$ month	$8^{th}/9^{th}$ month
Michele	Web tier	Web Client	Mobile client	Test Business
Mattia	Database	Business tier	Business tier	Test Clients
Alain	Business tier	Business tier	Business tier	Test Database

Table 26: Resource allocation for Implementation.

Resource	**2016-09-01 to 2016-09-31**	
	1^{st} and 2^{nd} week	3^{rd} and 4^{th} week
Michele	Web tier - Business tier	Business tier - Data
Mattia	Web tier - Business tier	Business tier - Data
Alain	Web Client - Web tier	Web tier - Mobile Client

Table 27: 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:** Studying previous works on a similar subject can be very helpful in this.
- **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, in order to squeeze even the tiniest time slots available for this project.
- **Risk:** A sudden growth in requirements can lead to a rush to meeting deadlines, jeopardizing the overall quality

  - **Probability:** Medium
  - **Damage:** Critical
  - **How to deal with it:** Thinking with a broader mind on the first stages can be very helpful; however, the team should be careful against over-engineering (which can also paralyze the development)

- **Risk:** Collaboration issues can sometimes be crucial, especially when dealing with task divisions.
  - **Probability:** Medium
  - **Damage:** Medium
  - **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 (3 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 almost surely slow down the entire team, which has to study these new technologies first
  - **Probability:** High
  - **Damage:** Critical
  - **How to deal with it:** This has to be account in the first stages of planning and inserted in the project scheduling.

- **Risk:** If the servers happen to be unreliable or in the case of more users than expected, a significant downtime can seriously damage the whole project
  - **Probability:** Medium

  - **Damage:** Critical
  - **How to deal with it:** A scalable design of the overall architecture is essential, both in software and in hardware choices.

- **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 has to be correctly verified and processed.

- **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 according to the specifications contained in [5]. A periodic check of requirements contained in [2] is also required.

5.3 Business Risks

- **Risk:** Testing devices & infrastructure (PCs, several mobile phones, server rent) 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 [2], in order to avoid worthless spendings.

- **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:** myTaxiService may violate some (future) laws related to taxi management.
  - **Probability:** Low
  - **Damage:** Critical
  - **How to deal with it:** A periodic check must be done in order to avoid legal consequences. In the case of drastic changes, the whole team must work in order to adapt to the new regulations as soon as possible.

- **Risk:** Another bigger company could acquire this company.
  - **Probability:** Low
  - **Damage:** Marginal
  - **How to deal with it:** No preventive solutions are available. Note that this is not necessarily a negative thing.

References

[1] Prof. Di Nitto - *Assignment 5 - Project Plan*

[2] Albanese Michele, Bianchi Mattia, Carlucci Alain - *myTaxiService: Requirements Analysis and Specification Document*

[3] Albanese Michele, Bianchi Mattia, Carlucci Alain - *myTaxiService: Design Document*

[4] Albanese Michele, Bianchi Mattia, Carlucci Alain - *myTaxiService: Code Inspection Document*

[5] Albanese Michele, Bianchi Mattia, Carlucci Alain - *myTaxiService: Integration Test Plan Document*

[6] Andrea Bignoli, Leonardo Cella - *MeteoCal - Project Report*

[7] Federico Migliavacca, Leonardo Orsello - *MeteoCal - Project Reporting*

[8] QSM - *Function Point Languages Table* (`http://www.qsm.com/resources/function-point-languages-table`)

[9] AA.VV - *COCOMO II - Model Definition Manual* (`http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf`)

[10] AA.VV - *COCOMO II - Drivers* (`http://sunset.usc.edu/research/COCOMOII/expert/cocomo/drivers.html`)