Politecnico di Milano

Scuola di Ingegneria Industriale e dell'Informazione

M.Sc. in Computer Science and Engineering

myTaxiService

Project Plan Document

Authors:

Angelo Gallarello

Edoardo Longo

Giacomo Locci

February 2, 2016

Version 1.0

Contents

List of Figures

1 Introduction

1.1 Revision History

Version	Changes
1.0	Creation of the document

1.2 Purpose and Scope

This document represents the *Project Plan*. Its aim is to privide an estimation of the project size, effort and cost. We also analyze the tasks and their schedule, together with the economical, technical and project risks.

1.3 List of Definitions and Abbreviations

Here we list all the the abbreviations that are used throughout this document:

- RASD (*Requirement Analysis and Specification Document*) of the *myTaxiService* system.
- DD (*Design Document*) of the *myTaxiService* system.
- ITPD (*Integration Test Plan Document*) of the *myTaxiService* system.
- PPD (*Project Plan Document*) this document.

1.4 List of Reference Documents

- COCOMO II Model Manual (`http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf`)
- FP MODELER (`http://www.functionpointmodeler.com/fpm-infocenter/index.jsp`)

2 Project size, effort and cost

2.1 Function Points Approach

This section tries to estimate the size of the software using the *function points* approach.
This approach is based on the concepts of :

- **External Input**: an elementary process in which data crosses the boundary from outside to inside.
- **External Output**: an elementary process in which derived data passes across the boundary from inside to outside
- **External Inquiry**: an elementary process with both input and output components that result in data retrieval from one or more internal logical files and external interface files
- **Internal Logical Files**: a user identifiable group of logically related data that resides entirely within the applications boundary and is maintained through external inputs
- **External Logical Files**: a user identifiable group of logically related data that is used for reference purposes only.

Data elements are counted following the *Function Point Modeler* [1] guide.
It's important to underline that the following estimation in done base on the assumption that we will have to develop **the Client apps** only for the Android Platform.

2.1.1 Function Points Tables

Table 1: Internal and External Logical Files Weights

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

[1] `http://www.functionpointmodeler.com/fpm-infocenter/index.jsp`

Table 2: External Input

	Data Elements		
Files Type Referenced	**1-4**	**5-15**	**16+**
1-2	Low	Low	Avg.
2	Low	Avg.	High
2+	Avg.	High	High

Table 3: External Output and External Inquiry Weights

	Data Elements		
Files Type Referenced	**1-5**	**6-19**	**19+**
2	Low	Low	Avg.
2-3	Low	Avg.	High
3+	Avg.	High	High

Table 4: UFP Complexity Weights

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

2.1.2 External Input

- User sign-up (from *User Client* to *Backend*)
- User login (from *User Client* to *Backend*)
- User data update (from *User Client* to *Backend*)
- User ride request (from *User Client* to *Backend*)
- User reservation request (from *Driver Client* to *Backend*)
- Taxi driver response (from *Driver Client* to *Backend*)
- Taxi driver login (from *Driver Client* to *Backend*)
- Taxi driver problem report (from *Driver Client* to *Backend*)

2.1.3 External Outputs

- Ride request notification (from *Backend* to *Driver Client*)
- Ride sharing notification (from *Backend* to *User Client*)
- Ride cost estimation (from *Backend* to *User Client*)

2.1.4 External Inquiry

- Ride sharing request

2.1.5 Internal Logical Files

- User
- Taxi Driver
- Sigle Ride
- Shared Ride
- Taxi Queue

2.1.6 External Logical Files

- Google Maps (from Google Maps API)

2.1.7 Unadjusted Function Points

In this section we calculate the UFP for our system, using a reference the tables provided in section Function Points Tables

Table 5: Internal and External Logical Files Weights

Function	**Weight**	**Points**
External Input		
User sign-up	LOW	3
User login	LOW	3
User data update	LOW	3
User ride request	AVG.	4
User reservation request	AVG.	4
Taxi driver response	LOW	3
Taxi driver login	LOW	3
Taxi driver problem report	LOW	3
External Outputs		
Ride request response	AVG	5
Ride sharing response	AVG	5
Ride cost estimation	LOW	4
External Inquiry		
Ride sharing request	AVG	4
Internal Logical Files		
User	AVG	10
Taxi Driver	AVG	10
Shared Ride	AVG	10
Taxi Queue	AVG	10
External Logical Files		
Google Maps	AVG	7

Table 6: UFP Total

Function Type	**Points**
External Input	26
External Outputs	14
External Inquiry	4
Internal Logical Files	40
External Logical Files	7
TOTAL	**91 UFP**

2.2 COCOMO II Approach

2.2.1 Effort estimation model

The *COCOMO II* model expresses **effort** as *PERSON-MONTHS*.
In particular, to estimate the total **effort** the following formula is used :

$$\text{Effort} = A \times SIZE^E \times \prod_i EM_i \tag{1}$$

where :

- A is given statistically and is equal to 2.94
- $SIZE$ is the size of the software expressed in KLOC
- E is an aggregation of five scale factors (SF) (retrieved in the *Scale factors estimation* section)
- EM are the *effort multipliers* of the *cost drivers* (retrieved in the *Cost drivers effort multipliers estimation* section)

In the following sections all the parameters are calculated to generate the final result of the formula.

2.2.2 Scale factors estimation

This section provides the estimation for the scale factors.

Name	**Factor**	**Value**
Precedentedness	Nominal	3.72
Development flexibility	Nominal	3.04
Risk resolution	High	2.83
Team cohesion	Very High	1.10
Process maturity	High	3.12
Total	$E = 0.91 + 0.01 \times \sum_i SF_i$	1.0481

Table 7: Scale Drivers estimations

2.2.3 Cost drivers effort multipliers estimation

This section provides the estimation for the effort multipliers of the cost drivers.

C_i	**Name**	**Factor**	**Value**
RELY	Required Software Reliability	Low	0.92
DATA	Data base size	Low	0.90
CPLX	Product Complexity	Nominal	1.00
RUSE	Required Reusability	High	1.07
DOCU	Documentation match to life-cycle needs	High	1.11
TIME	Execution Time Constraint	Nominal	1.00
STOR	Main Storage Constraint	Nominal	1.00
PVOL	Platform Volatility	Low	0.87
ACAP	Analyst Capability	High	0.85
PCAP	Programmer Capability	Nominal	1.00
APEX	Application Experience	Very low	1.22
PLEX	Platform Experience	Very low	1.19
LTEX	Language and Tool Experience	Low	1.09
PCON	Personnel Continuity	Very high	0.81
TOOL	Usage of Software Tools	Nominal	1.00
SITE	Multisite Development	High	0.93
SCED	Required Development Schedule	High	1.00
Total	$EM = \prod_i C_i$		0.795

Table 8: Effort multipliers estimation

2.2.4 Final effort estimation

Given :

- $A = 2.94$
- $SIZE = 91UFP \times 53 = 4.823KLOC$ (53 is the JAVA multiplier)
- $\prod_i EM_i = 0.795$
- $E = 1.0481$

$$\text{Effort} = A \times SIZE^E \times \prod_i EM_i = 12.15PM \tag{2}$$

The effort to develop the project is 12.15 person-months.
Given that we are 3 people this means a 4 months development.

2.2.5 Time to develop estimation

We calculate the *TDEV* parameter that *"for the waterfall model goes from the determination of a product's requirements baseline to the completion of an acceptance activity certifying that the product satisfies its requirements."*[2]

$$\text{Duration} = 3.67 \times (Effort)^{0.28+0.2\times(E-0.91)} = 7.91months \tag{3}$$

[2] `http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf`

3 Tasks

The fulfillment of the project requires a list of tasks to be completed in the order they are presented here, it is important to note, however that this list of tasks is intended to be used as a general guideline and all the tasks may receive updates along the way. Small changes can be made to the order, especially the last two and the first two steps that will be re-iterated over and over as new requirements emerge. So, the first version of each task will have strict deadlines, so that the following task can start, while the re-iterations will have specific deadlines that will be specified along the process.

Here is the a table that summirises the main tasks to be completed:

Order	**Task**
1	RASD
2	DD
3	ITPD
4	PPD
5	Implementation & Unit Testing
6	Integration Testing

Here we explain what each task is for:

1. Prepare a document that specifies the goals of the system, the assumptions that had been made, and the requirements of the *system-to-be*, namely the RASD.

2. Prepare a document that specifies the design and architecture of the *system-to-be*, namely the DD.

3. Prepare a document that specifies the plan to follow in order to perform the integration testing, namely the ITPD.

4. Prepare a document that specifies the plan of the project, namely the PPD.

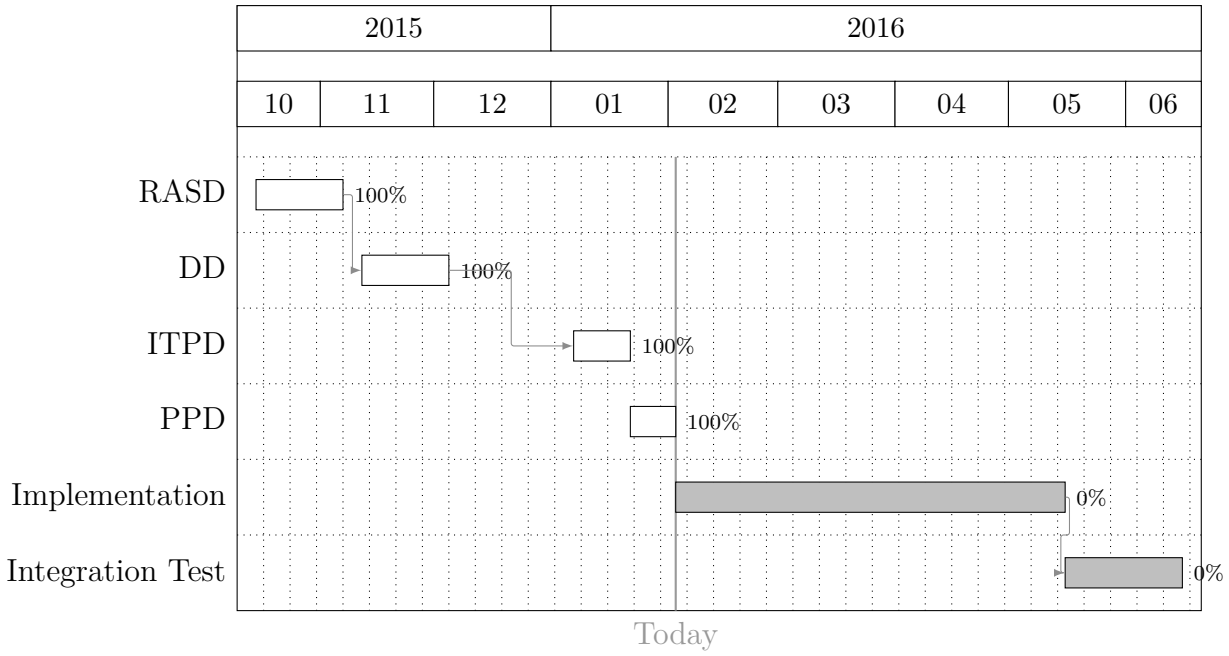
5. Write the software, according to what has been specified in the documents produces during the previous tasks, along with the unit testing.

6. Perform the integration testing, following the plan described in the ITPD.

3.1 Schedule

In this subsection we present a schedule that must followed, however, as always, small changes can always be made.

We explain the schedule through the Gantt chart that follows:



Since the precise deadlines are not legible from the Gantt chart, we give also a table summarising the main deadlines.

Order	**Begin date**	**Task**	**Deadline**
1	15-10-2015	RASD	06-11-2015
2	12-11-2015	DD	04-12-2015
3	07-01-2016	ITPD	21-01-2016
4	22-01-2016	PPD	02-02-2016
5	03-02-2016	Implementation & Unit Testing	15-05-2016
6	16-05-2016	Integration Testing	15-06-2016

4 Resources

In this section we specifiy the resource allocation, that is, for each task we specify to whom its subtasks are allocated. The timetables have already been defined in section 3.1. We choose to represent this data by means of tables. As far as the documents are concerned, we specify the resources for each section of the document, whereas, for what concerns the *Implementation + Unit Testing* we provide a resource for each of the three main applications that will have to be developed. Of course also this allocation can be subject to small changes in the future.

RASD

Subtask	**Resource**
Introduction	Edoardo
Overall Description	Angelo Edoardo
Specific Requirements	Angelo Giacomo
Appendices	Angelo Giacomo Edoardo

DD

Subtask	**Resource**
Overview	Edoardo
High Level Components	Angelo
Component View	Angelo Giacomo
Deployment View	Angelo Edoardo
Component Interfaces	Angelo
Architectural Styles and Patterns	Angelo Giacomo Edoardo
Data Management View	Edoardo
Algorithm Design	Giacomo
User Interface View	Angelo
Requirements Traceability	Angelo Giacomo Edoardo
Appendices	Angelo Giacomo Edoardo

ITPD

Subtask	**Resource**
Introduction	Edoardo
Entry Criteria	Angelo Edoardo
Individual Steps	Angelo Giacomo
Tools and Test Equipment Required	Edoardo
Program Stubs and Test Data Required	Angelo Giacomo Edoardo
Appendices	Angelo Giacomo Edoardo

PPD

Subtask	**Resource**
Introduction	Edoardo
Project Size, Effor and Cost	Angelo Giacomo
Tasks	Edoardo
Resources	Edoardo
Program Stubs and Test Data Required	Angelo Giacomo
Appendices	Angelo Giacomo Edoardo

Implementation + Unit Testing

Subtask	**Resource**
User Client	Edoardo
Driver Client	Giacomo
Server	Angelo

Integration Testing

Subtask	**Resource**
User Client Submodules	Edoardo
User Client →Server	Edoardo
Driver Client Submodules	Giacomo
Driver Client →Server	Giacomo
Server Submodules	Angelo

5 Risks

This project is directly linked with the risks that are described below.

5.1 Project risks

Delays The development of this project could slow down or stop for a period for reasons that are external of this company and we may not have any control on them. In case of delays we will release an Alpha/Beta version and then we will continue the development using an Extreme Programming modality.

Communication Issues Developer and designer will often work remotely. This can cause some coomunication issue or misunderstanding, to avoid them we will use a strict and clear communication pattern applied with tools created for this kind of situation. (i.e. Slack, Trello ecc.)

Lack of programming experience with specific framework The team has no experience in programming using JavaEE, this could cause some delays.

Requirement changes Requirements could change while the team is developing. Writing a clean, reusable and well structured code will be fundamental to face this risk.

Risk	**Probability**	**Effects**
Delays	Moderate	Moderate
Communication Issues	Very low	Moderate
Lack of experience	Certain	Low
Requirements change	Low	Low

Table 9: Project risks.

5.2 Technical risks

System fail System could go down for a period for technical reason, a emergency cluster of servers will guarantee that the service will just slow down avoiding the complete interruption.

Bad written code The code can become less usable and extensible with time, a good Design Document and a periodic code review will reduce this risk.

Data loss Hardware or software issue may cause a data loss. It will be very important to have a reliable back up plan to reduce the effect of this eventuality.

Data leaks Users data may be leaked with some deliberate attack to our servers. Testing security and cripting data movements will reduce drastically this risk.

Risk	**Probability**	**Effects**
System fail	Low	High
Bad written code	Very low	Catastrophic
Data loss	Very low	High
Data leaks	Very low	Catastrophic

Table 10: Technical risks.

5.3 Economical risks

Bankruptcy The all project could fail if the costs of maintenance and development would overcome earnings. That's why will be fundamental applying a strictly pessimistic feasibility study.

Competitors Other companies could realease a more reliable, feature and well designed software overcoming us. That's why it will be necessary to continue development even when the product will be released.

Risk	**Probability**	**Effects**
Bankruptcy	Very low	Catastrophic
Competitors	Low	Moderate

Table 11: Economical risks.

Appendices

A Tools

- *Sublime Text 2* as editor
- *LatexTools for Sublime Text 2 + MacTex* to build
- *Trello* for team coordination
- *Git + Git Flow* for version control

B Hours of work

- Angelo Gallarello : 7 hours
- Edoardo Longo : 7 hours
- Giacomo Locci : 7 hours