Assignment 5: Project Reporting

Matteo Rizzi (789467) & Claudio Scandella (853781)

February 2, 2016

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to evaluate the efforts and costs necessary to develop the MyTaxiService application. We do this by using two different types of costing metrics: the Function Points and the COCOMO.

The first is used to estimate the dimension of our project. The second one is used to compare the efforts to develop the application with the time really spent in doing it.

1.2 Definitions, Abbreviations and Acronyms

1.2.1 Definitions

- Function Points: a technique to assess the effort needed to design and develop custom software applications;
- COCOMO: The Constructive Cost Model (COCOMO) is an algorithmic software cost estimation model developed by Barry W. Boehm. The model uses a basic regression formula with parameters that are derived from historical project data and current as well as future project characteristics;
- Internal Logical File: homogeneous set of data used and managed by the application;
- External Interface File: homogeneous set of data used by the application but generated and maintained by other applications;
- External Input: Elementary operation to elaborate data coming form the external environment;
- External Output: Elementary operation that generates data for the external environment;
- External Inquiry: Elementary operation that involves input and output.

1.2.2 Abbreviations

- ILF: Internal Logical File;
- EIF: External Interface File;
- EI: External Input;
- EO: External Output;
- EIQ: External Inquiry;
- UFP: Un-Adjusted Function Points;

- RASD: Requirement Analysis and Specification Document;
- GPS: Global Positioning System;
- TD: Taxi Driver;
- FP: Function Point.
- GUI: Graphical User Interface

1.2.3 Acronyms

- COCOMO: COnstructive COst MOdel;
- SLOC: Source Lines Of Code.

1.3 References

- Project Plan Assignment.pdf;
- RASD version 1.1.pdf;

1.4 Overview

This document is divided in four different sections:

- Introduction: in this section we introduced the purpose of the Project Report document and some important informations in order to better understand what will be written in the following sections;
- Function Points Approach: in this section we evaluate thanks to the Function Points Approach the dimension of the project;
- COCOMO Approach: in this section we compare the efforts necessary to develop the application with the real time spent on doing it with the COCOMO Approach;
- Tasks and Schedule: in this section is shown major project tasks and how they are sheduled over the months;
- Resource Allocations: in this section every tasks is associated to at least a group member;
- Risk Analysis: in this section risks of the project are considered and a solution strategy is explained to each of them;
- Conclusion: in this section we list the tools used to redact the document and the time spent by each developer on writing it.

2 Function Points Approach

The Function Points Approach is used to evaluate the dimension of our project basing on the functionalities we identified in the RASD document. Once we obtained all the functionalities from the RASD we evaluate the cost and the complexity of each one of the functions with the help of a particular table, that we report below, that indicates the number of FPs basing on the single functionality and its complexity.

Functionality	Complexity		
	Simple	Medium	Complex
ILF	7	10	15
EIF	5	7	10
EI	3	4	6
EO	4	5	7
EIQ	3	4	6

2.1 Internal Logical File

The application includes a number of ILFs that are used to store data regarding different data types. The ILFs that we have individuated regards these entities:

- User;
- Guest;
- Taxi Driver;
- Taxi;
- Queue;
- Zone;
- Request;
- Reservation.

We considered all these entities as simple data because they are composed of fields that are simple, except for the Queue entity that we considered as complex because our application has more than one queue, that may be represented as a list, an array or another data structure, in which every field of the structure is a data structure itself.

In the table below we summarize the IFLs found, their complexity and the FPs necessary.

ILF	Complexity	N°of IFLs identified * Complexity Weight	Functional Points
User	Simple	$1 * 7$	7
Guest	Simple	$1 * 7$	7
Taxi Driver	Simple	$1 * 7$	7
Taxi	Simple	$1 * 7$	7
Queue	Complex	$1 * 15$	15
Zone	Simple	$1 * 7$	7
Request	Simple	$1 * 7$	7
Reservation	Simple	$1 * 7$	7
TOTAL			64

2.2 External Interface File

The application includes only one EIF to manage the interaction between the application and the GPS. This EIF results in only one entity of our Class Diagram, the Position class, which is an entity with a simple structure.

Therefore we only need 1*5=5 FPs.

2.3 External Input

The application interacts with the users in order to allow them to do some operations:

- User Registration: this is a simple operation, so we decided to adopt the simple weight. So 1*3=3 FPs;
- Login/Logout: these are both simple operations, therefore we decided to use the simple weight. So 2*3=6 FPs;
- User requests a taxi: this is a complex operation because it involves five entities (User, Taxi Driver, Zone, Position, Request). Therefore we decided to use the complex weight. So 1*6=6 FPs;
- User reserves a taxi: this is a complex operation because it involves four entities (User, Zone, Position, Reservation). Therefore we decided to use the complex weight. So 1*6=6 FPs;
- User modifies a reservation: this is a complex operation because it involves four entities (User, Zone, Position, Reservation). Therefore we decided to use the complex weight. So 1*6=6 FPs;
- User deletes a reservation: this is a simple operation but it involves four entites (User, Zone, Position, Reservation). So we decided to use the medium weight. So 1*4=4 FPs;
- TD informs the system about his availability: this is a simple operation that involves only the TaxiDriver and the Queue entities. So we decided to use the simple weight. So 1*3=3 FPs;

- TD informs the system of the end of a client's ride: this is a simple operation that involves only the TaxiDriver and the User entities. So we decided to use the simple weight. So 1*3=3 FPs;
- TD confirms a call: this is a simple operation but we decided to use the medium weight because it involves a great number of entities (TaxiDriver, Queue, Request / Reservation, Zone). So 1*4=4 FPs;
- TD rejects a call: this is a simple operation but we decided to use the medium weight because it involves a great number of entities (TaxiDriver, Queue, Request/Reservation, Zone). So 1*4=4 FPs;
- TD reports a client: this is a complex operation because it involves four entities (User, TaxiDriver, Report, Request / Reservation). Therefore we decided to use the complex weight. So 1*6=6 FPs;

In the table below we have summarized every EIs with his complexity and his cost in terms of FPs:

EI	Complexity	Functional Points
User Registration	Simple	3
Login/Logout	Simple	6
User request a taxi	Complex	6
User reserve a taxi	Complex	6
User modify a reservation	Complex	6
User deletes a reservation	Medium	4
TD informs the system about his availability	Simple	3
TD informs the system of the end of a ride for a client	Simple	3
TD confirms a call	Medium	4
TD rejects a call	Medium	4
TD reports a client	Complex	6
TOTAL		51

2.4 External Output

We identified some operations that generates data for the external environment:

- The system informs the TD about an incoming call: this is a complex operation because it involves these entities: User, TaxiDriver, Request / Reservation, Zone, Queue. So we decided to use the complex cost. Therefore 1*7=7;
- The system notify Users the waiting time and the Taxi Code for their Request / Reservation: this is a complex operation because it involves these entities: User, TaxiDriver, Request / Reservation, Notification. So we decided to use the complex cost. Therefore 1*7=7;

In the table below we have summarized all EOs we identified:

EO	Complexity	Functional Points
The system informs the TD about an incoming call	Complex	7
The system notify Users the waiting time and the Taxi Code	Complex	7
TOTAL		14

2.5 External Inquiry

We identified allows Users to request informations about the positions of other taxi through a map visualization. So:

- Allow taxi drivers to see the distribution of other taxis in the city in order to spread themselves uniformly: we considered this as a medium complexity operation because it involves three entities: the Taxi Driver, the Zone and the Position but it is a simple operation. So we decided to adopt the medium cost. Therefore 1*4=4.

EIQ	Complexity	Functional Points
Allow taxi drivers to see the distribution of other taxi in the city	Medium	4
TOTAL		4

2.6 Summary

The table below summarize the number of FPs for every type of functionality

Functionality	Functional Points
ILF	64
EIF	5
EI	51
EO	14
EIQ	4
TOTAL	138

Thanks to this data we can now calculate the estimated lines of code of our application. We are doing so thanks to the SLOC parameter, which is equal to 46. So the estimated lines of code of our application are:

$$TOTAL\,FPs * SLOC\,PARAMETER = 138 * 46 = 6348\,SLOC$$

3 COCOMO Approach

3.1 Introduction

We can use the COCOMO approach in order to confront the effort necessary to develop this project with the real time spent on doing it. We do this using some parameters defined in the COCOMO II Model Definition Manual `http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf`

There are two types of parameters:

- Scale Drivers;
- Cost Drivers.

3.2 Scale Drivers

- Precedentedness: It reflects the previous experience that we had with this kind of projects. This is our first experience in this type of project, so we decided to set the value to low.
- Development flexibility: It reflects the degree of flexibility in the development process. We decided to set this value to high because the professor did not go in the details, but they made only general explanations.
- Risk Resolution: Reflects the extent of risk analysis carried out. We decided to put this value to high because we consider ourself to be able to eliminate all general risks.
- Team Cohesion: Reflects how well the development team know each other and work together. We put this value to high because we had some initial difficulty because it was the first time we worked together and we had to know our habits and discover each other's abilities and ideas. But after these initial problems our cohesion was very good.
- Process Maturity:This was evaluated around the 18 Key Process Area (KPAs) in the SEI Capability Model. Because of the goals were consistently achieved these values will be set to nominal, level 2.

This table summarize all the Scale Drivers and the total of points, associated to each scale driver, is found to be used later on.

Scale Driver	Level	Points
Precedentedness	LOW	4,96
Development Flexibility	HIGH	2,03
Risk Resolution	HIGH	2,83
Team Cohesion	HIGH	2,19
Process Maturity	NOMINAL	4,68
TOTAL		16,69

3.3 Cost Drivers

- Required Software Reliability: We set this value to low because we know that software failures can happen and bring to loss of important data.
- DataBase Size: We set this value to nominal because it is probable that the ratio between the DataBase dimension and the SLOC could be between 10 and 100.
- Product Complexity: We set this value to nominal according to the COCOMO II CPLX rating scale.
- Required Reusability: We set this value to high because there are some reusable components in our application.
- Documentation Match to Life-Cycle Needs: This parameter describes the relation between the provided documentation and the application requirements. We set this parameter to low.
- Execution Time Constraint: We set this value to nominal because it's the lowest relevant level in the table.
- Main Storage Constraint: This parameter represents the degree of main storage constraint. So we decided to set this value to nominal because it's the lowest relevant level in the table.
- Platform Volatility: We set this value to low because we considered that every platform we used for the development needs and receives an important update every year and minor changes every month. This platforms are the DBMS, the Operating System and the browser.
- Analyst Capability: We set the analyst capability to high because we dedicated a lot of effort trying to satisfy all the requirements and choosing a simple design for our application that is user-friendly.
- Programmer Capability: Since we have not really programmed anything (we don't consider writing Alloy code and algoritmhs in pseudo-code as programming). So we set this value to low.
- Application Experience: Since this is the first time we work in a project like this one we set this value to low.
- Platform Experience: Since we worked on platforms like databases and user interfaces last year in the Software Engineering 1 project we set this value to nominal.
- Language and Tool Experience: This parameter is set to low since we used some tools and languages for the first time in this project.
- Personnel Continuity: We set this value to very low since the time we had for developing the application is less than half a year.

- Usage of Software Tools: We set this value to nominal because we used some tools including Gitlab for the repository.
- Multisite Development: This parameter reflects how we handled the distribution of development over distance and multiple platforms. Since we used phones, e-mails and other ways to work at this project collaborating at distance, since we live in different cities but in the same country, we set this value to nominal.
- Required Development Schedule: We set this value to high since our efforts were correctly distributed over the time given for each assignment, except for the RASD assignment in which we worked a lot in the latter phases because we had some differences to settle regarding some goals.

This table summarize all Cost Drivers with the weight we chose and the correspondent value we used to determine later on the EAF value.

Cost Driver	Level	Points
Required Software Reliability	LOW	0.92
DataBase Size	NOMINAL	1.00
Product Complexity	NOMINAL	1.00
Required Reusability	HIGH	1.07
Documentation match to life-cycle needs	LOW	0.91
Execution Time Constraint	NOMINAL	1.00
Main Storage Constraint	NOMINAL	1.00
Platform Volatility	LOW	0.87
Analyst Capability	HIGH	0.85
Programmer Capability	LOW	0.88
Application Experience	LOW	1.10
Platform Experience	NOMINAL	1.00
Language and Tool Experience	LOW	1.09
Personnel Continuity	VERY LOW	1.29
Usage of Software Tools	NOMINAL	1.00
Multisite Development	NOMINAL	1.00
Required Development Schedule	HIGH	1.00
TOTAL		0.90

3.4 Effort Equation

We found the effort estimation measured in Person-Months(PM) with the following equation:

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

Where:

- A is a constant defined by the COCOMO II standards and it is equal to 2.94;
- EAF is the product of all cost drivers found in section 3.3;
- KSLOC is the number of Source Lines of code found in the Functional Points Section;
- E is an exponent derived from the Scale Drivers.

The E parameter is calculated with this formula:

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

and:

- B is a constant defined by the COCOMO standards and it is equal to 0.91;
- SD_i is the i-th scale driver in the table in the section 3.2.

So we found that EAF is equal to 0.90, KSLOC is equal to 5.888 and E is equal to:

$$E = 0.92 + 0.01 * 16.69 = 0.92 + 0.1669 = 1.0869$$

So we can now find the effort with the first equation of this paragraph:

$$Effort = 2.94 * 0.90 * 6.348^{1.0869} = 19.723\,PM$$

3.5 Duration Equation

We calculated the Duration (Schedule), which is expressed in months, with the following equation:

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

Where in this equation:

- The Effort is the one found in the previous sub-section;
- F is calculated with another equation.

F is calculated with this formula:

$$F = 0.28 + 0.2 * (E–B)$$

So, first, we found the F parameter:

$$F = 0.28 + 0.2 * (1.0869 - 0.92) = 0.28 + 0.033 = 0.313$$

Next we found the Duration:

$$Duration = 3.67 * 19.723^{0.313} = 9.33\,Months$$

3.6 Conclusion

Thanks to the previous sub-sections we can now find the number of people N required to develop this project:

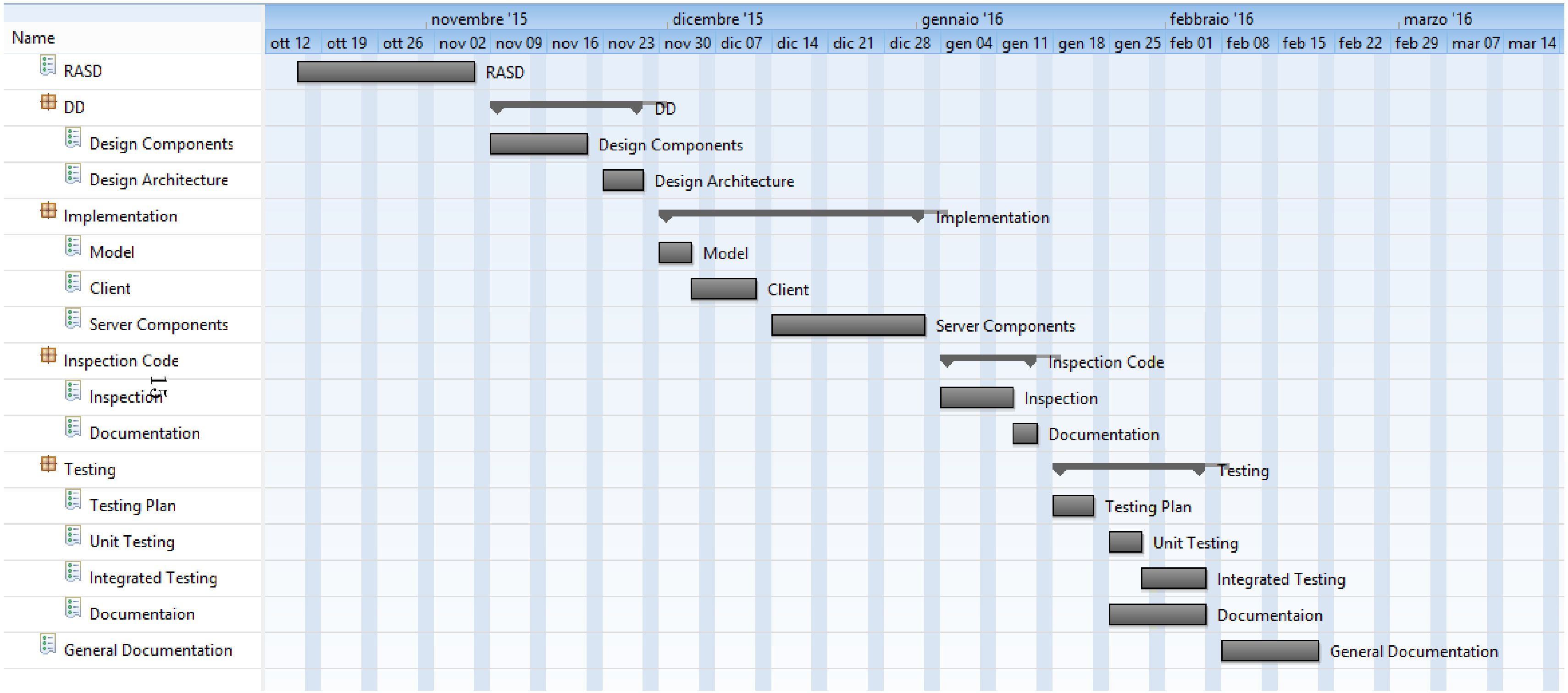
$$N = Effort/Duration = 19.723/9.33 = 2.11 \sim 2Person$$

4 Tasks and Schedule

Here is shown the project schedule with the most significant tasks: first in a table with time schedule and duration and then a schedule image shows the planning over the months.

In bold are the main tasks. In small font are the subtasks of the bold task just above them.

Task name	Duration	Starting time	Finish time
Project Presentation	**1 day**	**15-10-2015**	**15-10-2015**
RASD	**3 weeks**	**16-10-2015**	**6-11-2015**
DD	**3 weeks**	**9-11-2015**	**27-11-2015**
design components	2 weeks	9-11-2015	20-11-2015
design architecture	1 weeks	23-11-2015	27-11-2015
Implementation	**5 weeks**	**30-11-2015**	**1-1-2016**
Model	4 days	30-11-2015	03-12-2015
Client	6 days	04-12-2015	11-12-2015
Server Components	3 weeks	14-12-2015	01-01-2016
Inspection Code	**2 weeks**	**4-1-2016**	**15-1-2016**
Inspection	7 days	04-01-2016	12-01-2016
Documentation	3 days	13-01-2016	15-01-2016
Testing	**3 weeks**	**18-1-2016**	**5-2-2016**
Testing Plan	1 week	18-01-2016	22-01-2016
Unit Testing	4 days	25-01-2016	28-01-2016
Integrated Testing	6 days	29-01.2016	05-01-2016
Documentaion	2 weeks	25-01-2016	05-01-2016
General Documentation	**2 weeks**	**8-2-2016**	**19-2-2016**
Presentation	**1 day**	**2-3-2016**	**2-3-2016**



Name

RASD

DD

Design Components

Design Architecture

Implementation

Model

Client

Server Components

Inspection Code

Inspection

Documentation

Testing

Testing Plan

Unit Testing

Integrated Testing

Documentaion

General Documentation

5 Resource Allocations

In this section is defined the allocation of each component of the project. Rizzi Matteo (R) and Scandella Claudio (S) are the project team and they will cover all tasks, so they act as developers, designers and testers.

The following table shows the allocation to various tasks.

Task name	People
RASD	R - S
DD	R - S
design components	R - S
design architecture	R - S
Implementation	R - S
Model	R
Client	R
Server Components	S
Inspection Code	R
Inspection	R
Documentation	R
Testing	R - S
Testing Plan	R - S
Unit Testing	S
Integrated Testing	S
Documentaion	R
General Documentation	R - S

6 Risks Analysis

Risks are hidden everywhere in all type of project and can be sometimes difficult to find. MyTaxiService project is not an exception, so here are listed some possible risks as well as their probability that they can occur and impact that they will do if occur.

Risk N°	Risk	Probability	Effect
1	Schedule is not appropriate because tasks need more time to be performed.	High	Critical
2	Tasks allocation is not well defined so one of the group has more work to do with respect to the other one.	High	Serious
3	Requirements can change and so it must be modified the design of the app.	Moderate	Critical
4	Difficulty in implementing or designing some elements.	Moderate	Critical
5	Financial problems due to a force majeur event make government to cut the budget.	Moderate	Catastrophic
6	Some group's component can be ill.	High	Critical
7	Some unexpected event like breaking PC can delay the schedule.	Moderate	Serious

Level of effect is scaled, from the less critical to the most critical: Serious, Critical, Catastrophic.

The following table, for each risk, shows a solution to obviate them.

Risk N°	Solution strategy
1	Some schedule time of any tasks can be oversized so if it is performed in advance other tasks can be started before the projected time.
2	Person that finishes his task can join the other one helping him to quicken the project.
3	Prepare a document that explains how change in requirements can affect schedule time and project budget, inviting to not do huge change to them.
4	Prepare an alternative with reduced functionalities to simplify the implementation and/or the design, with a briefing document for the government that requested the application.
5	Reduce performance optimization and/or architectural component's quality and/or GUI quality to the extent possible.
6	People must be available to extrawork in order to not delay the schedule.
7	Substitutes PCs can be found in the university's study rooms.

7 Conclusion

7.1 Tools used

These are the tools we used to redact the Project Report Document:

- LyX: in order to write the document;

7.2 Hours of work

This is the amount of time each developer worked on the redaction of this document:

- Rizzi Matteo: 13 hours;
- Scandella Claudio: 13 hours;