Politecnico di Milano

Department of Electronics, Information and Bioengineering

Master Degree course in Computer Science Engineering

Project Plan (PD)

myTaxiService

Instructor: Prof. Elisabetta Di Nitto

Authors:	*Code:*
Luca Luciano Costanzo	789038
Simone Disabato	852863

Document version 1.0, released on January 25, 2016

Contents

Abstract

The global scope of the project follows, as it has been explained in the previous documents, to make the document easy to understand.
Users, once registered, are able to ask for an immediate ride or to book one of them.

The system provides the user with a complete map of the city and its suburbs within the taxi service is available. The current position of the user is obtained by localization services of the user's smartphone if it's possible, otherwise the user notifies his position directly on the map with a marker or by a searching box. The destination is also chosen either graphically or by a research. The user can view the suggested path and then he must confirm the request.
When a user asks for a ride, the system checks the availability of a taxi driver near the current position, by splitting the city in several areas and using a FIFO (First In First Out) policy to manage the assignment of the ride's driver. The selected driver can accept or decline the ride. In the former case the system informs the user about waiting time, estimated travelling time, prices and cab car-code.
The system gives also the possibility to book a ride with at least two hours in advance. As the user does when he asks for a ride, he selects the desired starting venue and the destination. Afterwards, the system gives a calendar where the customer can choose the date (at most 30 days in advance) and the starting hour. Ten minutes before the meeting time the system starts all the operations described before in order to assign a taxi-driver.
A reservation from the app or the website can be undone until the system confirmation of the availability of a taxi, while a booking can be cancelled at most fifteen minutes before the meeting hour.
After those deadlines the ride is considered bought by the customers and an eventual absence on the established venue forbids other possibilities to book or to take a ride.

Chapter 1

Introduction

In this chapter the purpose of the document will be presented in the section 1.1. Then, other useful information are made available, for instance the list of definitions and abbreviations and the reference documents. Finally, in the section 1.4 an overall description of the document structure will be presented.

1.1 Purpose

The purposes of this document are principally two. The first one is to estimate the project size, the effort and the cost, by using some algorithmic procedures. Second, a schedule and a plan for the document (partially retroactive, since this section should have been written in parallel with the Requirements Analysis and Specification Document), having a detailed analysis of team's member availability, the risks associated to the project and the associated recovery actions.

1.2 List of Definitions and Abbreviations

Up to now, no definitions or acronyms or abbreviation have been used in the document. Hence, this section is empty.

1.3 List of Reference Documents

The reference documents are now listed. Note that, all the documents related on the *myTaxyService* project are written by the same authors of this one, whereas the other documents have a reference of their author when this information is available.

- Software Engineering 2 Project, AA 2015-2016 Assignments 4 - Test plan (available on beep platform only for registered students of Politecnico of Milan);
- The Requirements Analysis and Specification Document (RASD) for *myTaxiService* - v1.2, released on 6th November 2015;
- The Design Document (DD) for *myTaxiService* - v1.0, released on 4th December 2015;
- The Integration Test Plan Document (ITPD) for *myTaxiService* - v1.01, released on 21th January 2016.
- COCOMO II Model Definition Manual, available at `http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf`

1.4 Overall Description

The estimations concerning the project are presented in the chapter 2, with two algorithmic techniques: the Function Points (FP) to (typically under-) evaluate the project size and the COnstructive COst MOdel (COCOMO) to estimate the project effort and the costs.
The chapter 3 is reserved to the project schedule presentation and to the assignment of each task to a project's developer.
Finally, the chapter 4 the risks of the project and the related actions will be presented.

Chapter 2

Project Estimations

In this chapter we are going to estimate the main features of *myTaxiService* project, by using COCOMO II. Reading from the reference manual:

> *The COCOMO II model is part of a suite of Constructive Cost Models. This suite is an effort to update and extend the well-known COCOMO (Constructive Cost Model) software cost estimation model originally published in Software Engineering Economics by Barry Boehm in 1981.*

In the section 2.1 we focus on the project's size in term of lines of code, whereas in the section 2.2 other metrics, such the required time and the costs will be analysed.

2.1 Project Size (Function Points)

Reading from the reference manual:

> *The function point cost estimation approach is based on the amount of functionality in a software project and a set of individual project factors [Behrens 1983; Kunkler 1985; IFPUG 1994]. Function points are useful estimators since they are based on information that is available early in the project life-cycle. A brief summary of function points and their calculation in support of COCOMO II follows.*

The function types are five, described in the table[1].

Function Point	Description
External Input (EI)	Count each unique user data or user control input type that enters the external boundary of the software system being measured.
External Output (EO)	Count each unique user data or control output type that leaves the external boundary of the software system being measured.
Internal Logical File (ILF)	Count each major logical group of user data or control information in the software system as a logical internal file type. Include each logical file (e.g., each logical group of data) that is generated, used, or maintained by the software system.
External Interface Files (EIF)	Files passed or shared between software systems should be counted as external interface file types within each system.
External Inquiry (EQ)	Count each unique input-output combination, where input causes and generates an immediate output, as an external inquiry type.

Finally, to perform the analysis we have to present other two tables from the same reference manual of the other one. The first one will be used to classify each function on three level of complexity (high, medium and low).

The second one shows the weights to be used into the estimation formulas[2].

[1] The table is given by the COCOMO II reference manual.
[2] The UFP acronym means Unadjusted Function Points

Table 2. FP Counting Weights

For Internal Logical Files and External Interface Files

Record Elements	Data 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

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

For External Input

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

Table 3. UFP Complexity Weights

Function Type	Complexity-Weight: 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

Up to now, we have presented the Function Points technique. Now, we are going to start our analysis, split by the function type.

2.1.1 Internal Logic Files

The system has to manage Internal Logic Files to store information related to the users (both *normal* and drivers), the *historical* rides, the areas and the driver work shifts[3]
The users have from 12 to 16 fields to be stored (the second number is referred to the drivers case) and only the alerts and the zerotime or future rides have to be stored, thus the complexity is low. The areas and the work shifts can also be considered as low complexity type. In fact they have a few fields and less than six extra records.
The rides have 10 fields, including two positions, the driver and the passenger, all saved in separate entities. They can be considered as an average complexity type, since we have about seven records per ride (in fact in addition to the five presented, the positions requires additional records).

In the table the analysis is summarize:

ILF	Complexity	FP
User	Low	7
Area	Low	7
Workshift	Low	7
Ride	Average	10
Total		31

2.1.2 External Logic Files

The system acquires data from te GPS interface. A GPS position is essentially a tuple of type Position, described in our database. Hence, we have a low complexity type and 5 FP.

[3]See the logic schema at the page 10 of the Design Document to have a detailed description of each part of the database.

2.1.3 External Inputs

The possible interactions between the users and the system, defined in the RASD, are now quickly described in terms of complexity:

- Login/Logout: these operations are simple due to one entity only is involved, so the complexity is low;
- Start Waiting Time/End of a ride: these operation requires to interact with three types of files (Position, Area and Driver Waiting) with one element per type, thus the complexity is low;
- Check the Reservations: this is a group of three related operations (shows the alerts and gives the possibilities to modify or to cancel a ride) that involve one type and potentially more than 16 elements, so the complexity is average;

In the following table we have summarized the results:

EI	Complexity	FP
Login/Logout	Low	2x3
Start Waiting Time / End of a Ride	Low	2x3
Check the Reservations	Average	3x4
Total		24

2.1.4 External Inquiries

The system allows the user to interact with it through the following operations:

- Registration: this operation is performed only by simple user (not a driver) and involves one data type, the one related to the new user. Its complexity is low;
- Profile Management: this operation allows the user to modify a little personal data, so the complexity is low;
- Work shift Management: this operation requires to interact to 2 entities (driver and work shift) and can involve more than 20 elements to perform the validity checks. Hence the complexity is high;
- Book a ride (both future or zerotime): these operations needs many inter-

actions between the system and the user, for instance to show the detected position or to analyse the inserted time/address;

- Ride Allocation: we also insert this function because the system has to interact in a simple way with the taxi driver. Since many and many taxi drivers or queues may be involved before one is available, the complexity is high. Note that the notifications of the operations are not reported here, but in the External Input section;
- Taxi Driver Ride Request: this is the operation used to ask a rider the availability for a ride, the complexity is low.

In the following table we have summarize the results:

EQ	Complexity	FP
Registration	Low	3
Profile Management	Low	3
Workshift Management	High	6
Book a ride	High	2x6
Ride Allocation	High	6
Taxi Driver Ride Request	Low	3
Total		33

2.1.5 External Outputs

The external outputs shown by the system are all related with the notifications. In fact the system administrators can notify all the users about service situations (for instance a strike, an incident that forbids the access in some city areas and so on). Other kind of notifications, are the ones about the requested ride's status.
Finally we have the operations of shown these notifications (Read the Alerts). All the described operations have to interact with about three types of record (always the user and the alert. If any, also other files are involved, as the ride or the position) and many files (both users or old alerts when we are showing the alerts), thus the complexity is high. Instead, the ride status notifications have to interact with one user, so the complexity is low.

The results are summarized in the following table:

EO	Complexity	FP
System Notifications	High	7
Ride Status Notifications	Low	4
Read The Alerts	High	7
Total		18

2.1.6 Final Results

The Function Point found in the previous sections are reported in the following table:

Function Type	Value
Internal Logic Files (ILF)	31
External Logic Files (ELF)	5
External Inputs (EI)	24
External Inquiries (EQ)	33
External Outputs (EO)	18
Total	111

Since our project has no a specific programming language to be used, in the following table we report the project size estimations both for the C++ and for the Java. In addition we report a few interesting measure with C, Assembler and Machine Code:

Programming Language	UFP to SLOC Conversion Ratios[4]	Lines of Code
Java	53	5833
C++	55	6105
C	128	14208
Assembly - Basic	320	35520
Machine Code	640	71040

[4]The values are still taken from the COCOMO II reference manual.

2.2 Effort Estimation (COCOMO II)

In this section we make an Effort Estimation using the model of COCOMO II. Following this model we can calculate the required effort measured in Person-Months with the effort equation, that is:

$$effort = 2.94 * EAF * (KSLOC)^E \tag{2.1}$$

where the first factor 2.94 is a constant calculated by COCOMO II 2000.0, **EAF** is the *Effort Adjustment Factor* calculated as the product of the effort multipliers corresponding to each of the *Cost Drivers*, **KSLOC** is the estimated number of lines of code calculated at the end of section 2.1 and measured in thousands, the exponent **E** is the exponent derived from the *Scale Drivers*.

First we calculate the *Scale Drivers* in the first subsection and then the *Cost Drivers*. Finally we will compute the result of the effort equation presented above and also the duration of the necessary work to complete the project and the number of required people.

2.2.1 Scale Drivers

The exponent **E** in equation 2.1 is an aggregation of five scale factors (SF) that account for the relative economies or diseconomies of scale encountered for software projects of different sizes. These five scale factors with their corresponding numerical values are represented in the Table 2.1

Table 2.1: Scale Factor Values, SF_j, for COCOMO II Models

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	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

The scale factors and their values for our project are:

- **PREC - Precedentedness:**
  Reflects the previous experience of the organisation with this type of project. Since we haven't realised before this kind of commercial, distributed and multi-platform system (accessible both via website and smartphone), the value of this scale factor is *Low* (4.96).
- **FLEX - Development Flexibility:**
  Reflects the degree of flexibility in the development process. The instructor has set only some general requirements and goals without going in detail, hence the development of the project is very flexible. The value of this scale factor is *Very High* (1.01).

- **RESL - Architecture/Risk Resolution:**
  Reflects the extend of risk analysis carried out. We have studied the necessary architecture for the system and we have identified and prevented many critical security risks, hence this value is *High* (2.83).
- **TEAM - Team Cohesion:**
  Reflects how well the development team know each other and work together. In our case we know each other very well and we have also worked together before for another important project. For the development of myTaxiService we have worked together most of the time and other times we have split equally the work between us and worked independently at home, always communicating. Thus this value is *Extra High* (0.00).
- **PMAT - Process Maturity:**
  Reflects the process maturity of the organisation. TO BE CONTINUED

The results are resumed below:

Scale Driver	**Factor**	**Value**
PREC - Precedentedness	Low	4.96
FLEX - Development Flexibility	Very High	1.01
RESL - Architecture/Risk Resolution	High	2.83
TEAM - Team Cohesion	Extra High	0.00
PMAT - Process Maturity	TODO	TODO
Total		8.8 TO UPDATE

2.2.2 Cost Drivers

Chapter 3

Project Schedule

This chapter is dedicated to the project schedule. For each part of the project we report in a table the deadline, (not necessarily) a probably initial date, the description and the division among the team members (who are the authors of this document), identified by the following characters: '**L**' for Luca Luciano Costanzo and '**S**' for Simone Disabato.
To clarify the deadlines, we suppose that we have at our disposal about 20-25 hours per week to work at the project development[1].

Initial Date	Deadline	Worker(s)	Description
October 2015	6th November 2015	L , S	Requirements Analysis and Specification Document (RASD) and Project Planning[2].
7th November 2015	4th December 2015	L, S	Design Document (DD).
4th December 2015	6th January 2016	L , S	Working on other assignments.
6th January 2016	18th January 2016	L , S	Integration Test Plan Document (ITPD).

[1]Note that the value 25 hours per week is the same used in the section 2.2 to calculate the required person-month.

[2]In a real project the Project Plan should be written in parallel with the RASD.

Initial Date	Deadline	Worker(s)	Description
	3rd March 2016	L , S	Politecnico Exams
4th March 2016	to be defined (value A)	L	Development of the Database structure and of the DataBase Management Service.
4th March 2016	to be defined (value B < A)	S	Development of the part of the System controller concerning the users (creation and identification)
B	to be defined (value C A?)	S	Development of the System Security Manager.
C = A	D	L , S	Unit Tests of the part developed by the other team member. Application of the Integration Tests available on the developed components. Correction of code errors.
D	E	L , S	Development of the Client and User Handler, with Unit Test in parallel.

Chapter 4

Project Risks

In this chapter we are going to discuss about the risks associated with our project and for each of them we will provide a detailed analysis and some possible recovery actions.

4.1 Taxi Position Identification

The most critical component of the system is the queue management and the taxi position identification. Since these services are provided by external components (i.e., the GPS and the Google Maps API), we cannot fix eventual problems related to their functionalities' working.
Supposing that these components are fully tested and we can guarantee the correct provided results in each relevant situation (for our application), the risk is still present.
In fact, the GPS installed on each taxi has a reliability not equal to 1 (as each real component): when a fault occurs the system should be able to correctly identify the taxi and to allow the driver to continue his work. This problematic has been considered during Design phase: a driver can insert manually his position and continue to notify the system about his availability for a ride and his current area. Obviously, the case in which the taxi driver inserts a wrong address to make a joke is not considered because it is not realistic: each driver wants to

work and to earn the commissions guaranteed by the rides.

4.2 Internet connection

The system works through Internet, so we have ensured (even if we did not write) that each person (both user and driver) who is using the system is connected to Internet with a stable connection, thus the connection cannot go down for a long time.
In the application working this assumption is not always true and it assumes particular relevance when the involved person is a driver. Since he cannot communicate with the system, the latter one is no longer able to manage correctly his position and to assign him a ride. The problem may be related to several causes and most of them are not related to our application. For instance, some phone companies have a bad coverage on the city and the driver has a contract with those companies. This problem we suppose that is solved directly by the cab company which have equipped all its drivers with a work-phone having a connection provided by a good company.

4.3 Taxi accidents

The system we have designed has no way to detect the accidents. If the accident occurs during a ride, the driver has to notify the company by a call (about the gravity of the accidents, the passenger's health conditions, the assurance requested data and so on), but not the system which still considers the driver performing a ride.
If the accident occurs during the transfer to the starting position of a ride, the driver also call the company to notify the event (providing the usual information). The company takes charge of the situation, hence it will call the passenger to inform it and (the administrators) will notify the system about the "lost" driver and the ride which needs a new cab-man.

Chapter 5

Other Info

This chapter contains information about the used tools and the hours of work by the members of the working group.

5.1 Working hours

Date	Costanzo's hours	Disabato's hours
2016/1/21	-	3h
2016/1/22	-	3h
2016/1/23	4h	-
Total Project Plan	4h	6h
Global	75h	79h

5.2 Tools

For this assignment the following tools were used:

- LaTeX and TexStudio editor