

**POLYTECHNIC UNIVERSITY OF MILAN**

School of Industrial and Information Engineering

Computer Science and Engineering



**Project of Software Engineering 2:**  
**MyTaxi Service**  
**Project Plan**

Course Professor: Prof. Elisabetta DI NITTO

Authors:

Mattia CRIPPA 854126

Francesca GALLUZZI 788328

Marco LATTARULO 841399

Academic Year 2015–2016

# Contents

1	Introduction	2
2	Function Points Approach	3
3	COCOMO II Approach	7
4	Project Schedule & Resources Allocation	10
5	Project Risks	11

# Chapter 1

## Introduction

In this document we are going to evaluate the dimension of the code of the entire project and the time and effort needed by our team to realize it. For the estimation of the code size we have used the Functional Point approach. For the evaluation of effort and time needed we have used the COCOMO II model.

## Chapter 2

# Function Points Approach

The Function Point approach is a technique that allows to evaluate the total dimension of the program and therefore the effort needed to develop the software product depending on its functionalities. The list of functionalities has been obtained from the RASD and from the Design Document. There are 5 types of Functional Points:

- **Internal Logic File (ILF)**: a homogeneous set of data handled by the system.
- **External Interface File (ELF)**: a homogeneous set of data used by the application but handled by external application.
- **External Input**: elementary operation to elaborate data coming from the external environment.
- **External Output**: elementary operation that generates data for the external environment that usually includes also the elaboration of data from logic files
- **External Inquiry**: elementary operation that involves both input and output operations done without significant elaboration of data from logic files.

Function Types	Simple	Medium	Complex
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6
Internal Logical Files	7	10	15
External Logical Files	5	7	10

Using the Function Points method and the table of weights above, we can calculate the FPs of the entire system as the sum of the FPs obtained by evaluating every of the 5 types of Functional Points. In particular we are going to define a number of items for every type of FP and assign them a weight representing its complexity.

- **Internal Logic File (ILF):** The application includes a number of ILFs that will be used to store the information about *passenger*, *taxi drivers*, *rides*, *routes and reservation / request calls*. Passenger and Taxi Driver entities have a simple structure as they are composed of a small number of fields; Ride and Route entities have a medium complex structure; Call entity instead has a very complex structure. Thus, we can decide to adopt complex weight for call, medium weight for ride and route and simple weight for passenger and taxi driver entities. This means that we will come out with  $2 * 7 + 1 * 10 + 1 * 15 = 49$  FPs concerning ILFs.
- **External Interface File (ELF):** The application features also only one EIFs to manage the interaction with maps APIs. This information results in only one entity with a complex structure. Thus, we decide to adopt a complex weight. As a result, we get  $1 * 10 = 10$  FPs.
- **External Input:** The application interacts with the user to allow him/her to:
  - Login / Logout / Sign Up: these are simple operations, so we can adopt the simple weight for them  $3 * 3 = 9$  FPs.
  - Reserve / Request a taxi: these are not a simple operation, so we can adopt the medium weight  $2 * 4 = 8$  FPs.

- Update Account: this is a simple operation so we can adopt a simple weight  $1 * 3 = 3$  FPs.
- Signal Availability: this is a simple operation so we can adopt a simple weight  $1 * 3 = 3$  FPs.
- Accept or Decline Call: this is a simple operation so we can adopt a simple weight  $1 * 3 = 3$  FPs.
- Insert Testing Code: this is a simple operation so we can adopt a simple weight  $1 * 3 = 3$  FPs.

As a result, we get  $7 * 3 + 2 * 4 = 29$  FPs.

• **External Output:**

- After specific request, application will provides information about incoming taxi to passenger: this operation is not too complex so we can adopt medium cost  $1 * 5 = 5$  FPs.
- At the end of a ride, application will provides receipt: this operation is not too complex so we can adopt medium cost  $1 * 5 = 5$  FPs.
- After specific request, application will delivers incoming requests to taxi drivers: this operation is simple so we can adopt simple cost  $1 * 4 = 4$  FPs.
- After specific request, application allows taxi drivers to visualize information about optimal route to taxi drivers: this operation is quite complex so we can adopt complex cost  $1 * 7 = 7$  FPs.
- After specific request, application allows taxi drivers to visualize the dashboard: this operation is not too complex so we can adopt medium cost  $1 * 5 = 5$  FPs.

As a result, we get  $4 + 3 * 5 + 7 = 26$  FPs.

• **External Inquiry:** The application allows user to request information events, in particular user can see:

- Information about the system if he is a developer: this operation is simple so we can adopt simple cost  $1 * 3 = 3$  FPs.

- The summary of receipts if he is a passenger: this operation is simple so we can adopt simple cost  $1 * 3 = 3$  FPs.
- The summary of rides if he is a taxi driver: this operation is simple so we can adopt simple cost  $1 * 3 = 3$  FPs.

As a result, we get  $3 * 3 = 9$  FPs.

In summary, we have computed the following value for the unadjusted FPs: 123. This value can be used directly to estimate the effort in case we have some historical data that tell us how much time we usually take for developing a FP. Otherwise, it can be used as a basis to estimate the size of the project in KLOC and then use another approach such as COCOMO to estimate the effort. We can use the standard conversion coefficient to compute the number of Source Lines Of Code (SLOC) starting from the FPs. Therefore:

$$SLOC = 46 * FPs = 5658$$

## Chapter 3

# COCOMO II Approach

For carrying out this section of the document we are referring to the COCOMO II model definition manual at:

[http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII\\_modelman2000.0.pdf](http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf).

Following this document we have compiled the two tables below, and then we have used this (<http://csse.usc.edu/tools/COCOMOII.php>) online calculator for computing the effort, the total cost, the duration and the number of person estimated for the completion of the project.

Scale Driver	Factor	Value
Precedentedness	Very Low	6,20
Development Flexibility	Nominal	3,04
Risk Resolution	Low	5,65
Team Cohesion	Extra High	0
Process Maturity	High	3,12



Cost Driver	Factor	Value
Required Software Reliability	Nominal	1
Data Base Size	Nominal	1
Product Complexity	Nominal	1
Required Reusability	High	1,07
Documentation match to life-cycle needs	High	1
Execution Time Constraint	Nominal	1
Main Storage Constraint	Nominal	1
Platform Volatility	Nominal	1
Analyst Capability	Low	1,19
Programmer Capability	Nominal	1
Application Experience	Low	1,10
Platform Experience	Low	1,09
Language and Tool Experience	Nominal	1
Personnel Continuity	Very Low	1,29
Usage of Software Tools	Nominal	1
Multisite development	Low	0,86
Required Development Schedule	Nominal	1


So, in conclusion we have found these values for effort, total cost, duration and members of the project:

**Effort** = 25,5 Person-Months

**Cost** (assuming an average cost per Person-Month of 2.000,00 \$) = 51.062,00 \$

**Duration** = 10,7 months

**Team size** =  $25,5/10,7 = 2,38$  members



**COCOMO II - Constructive Cost Model**

Model(s)   
 COCOMO   
 Monte Carlo Risk ☐ Off   
 Auto Calculate ☐ Off

**Software Size**      Sizing Method

SLOC      % Design Modified      % Code Modified      % Integration Required      Assessment and Assimilation (0% - 8%)      Software Understanding (0% - 50%)      Unfamiliarity (0-1)

New

Reused

Modified

**Software Scale Drivers**

Precedentedness       Architecture / Risk Resolution       Process Maturity

Development Flexibility       Team Cohesion

**Software Cost Drivers**

**Product**

Required Software Reliability

Data Base Size

Product Complexity

Developed for Reusability

Documentation Match to Lifecycle Needs

**Personnel**

Analyst Capability

Programmer Capability

Personnel Continuity

Application Experience

Platform Experience

Language and Toolset Experience

**Platform**

Time Constraint

Storage Constraint

Platform Volatility

**Project**

Use of Software Tools

Multisite Development

Required Development Schedule

**Maintenance** ☐ Off

**Software Labor Rates**

Cost per Person-Month (Dollars)

---

**Results**

**Software Development (Elaboration and Construction)**

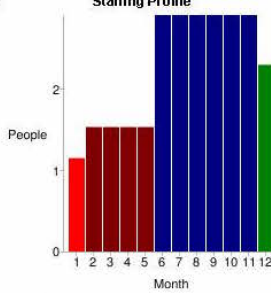
Effort = 25.5 Person-months  
 Schedule = 10.7 Months  
 Cost = \$51062

Total Equivalent Size = 5658 SLOC

**Acquisition Phase Distribution**

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	1.5	1.3	1.1	\$3064
Elaboration	6.1	4.0	1.5	\$12255
Construction	19.4	6.7	2.9	\$38808
Transition	3.1	1.3	2.3	\$6128

**Staffing Profile**



**Software Effort Distribution for RUP/MBASE (Person-Months)**

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.2	0.7	1.9	0.4
Environment/CM	0.2	0.5	1.0	0.2
Requirements	0.6	1.1	1.6	0.1
Design	0.3	2.2	3.1	0.1
Implementation	0.1	0.8	6.6	0.6
Assessment	0.1	0.6	4.7	0.7
Deployment	0.0	0.2	0.6	0.9

Your output file is [http://csse.usc.edu/tools/data/COCOMO\\_January\\_29\\_2016\\_09\\_35\\_09\\_57568.txt](http://csse.usc.edu/tools/data/COCOMO_January_29_2016_09_35_09_57568.txt)

Created by Ray Madachy at the Naval Postgraduate School. For more information contact him at [rjmadach@nps.edu](mailto:rjmadach@nps.edu)

Figure 3.1: Representation of effort, total cost, duration and number of person estimated for the completion of the project.

## Chapter 4

# Project Schedule & Resources Allocation

In defining the possible schedule for the project we have tried to adhere as much as possible to the result of the previous evaluation made using FP and COCOMO II adapting those prescriptions to our specific case (a team composed of 3 people). We have identified 6 major tasks to carry out during the development of the project:

- T1 : Feasibility study
- T2 : Analysis of Requirements and specifications
- T3 : Design of the system
- T4 : Coding and Unit testing
- T5 : Integration test and system test
- T6 : Deployment

In the effort of balancing as much as possible the workload of each component of the team we have developed the following Gantt diagram showing the distribution of every task among the team members during the months, forecasting a total duration of 10 months (rounding down with respect to the prescription obtained using COCOMO II to balance the dimension of the team - 3 people instead of 2,38).

## Chapter 5

# Project Risks

Given the nature of the project and the small dimension of the group of developers (3 person), the major risks are related to the availability of all the members of the team and to the respect of the time schedules. In fact, every member of the team has the same importance and has more or less the same skills of the other, so it's not possible to fully replace one team member in a short time, and it's also difficult for the two remaining members to carry out the scheduled tasks efficiently, in case of illness or unavailability of the third member. Also from the small number of developers depend all the possible risks related to unexpected modifications to the design of the system or also to the scheduling. The lack of hierarchy and the uniformity of skills lead to a necessity of continuous meetings and sharing of ideas and works. All this leads to small reactivity to changes and longer development time. One possible idea for solving or preventing such problems is to enlarge the team or at least to involve other people in some of the aspect of the project, and try to make all the documentation as much complete and effective as possible, so to make easier and faster for a new member the ingress in the team and the understanding of what's going on. Another possible idea is to better organize the distribution of the tasks in the team, trying to reach an optimal balance between collaboration and individual work. Trying to enforce collaboration only on the fundamental tasks and enforce decoupling on all the other minor tasks.