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

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 each one 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 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 only one EIF 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, the application will provide information about incoming taxi to passengers: this operation is not too complex so we can adopt a medium weight: $1 * 5 = 5$ FPs.

  - At the end of a ride, application will provide a receipt: this operation is not too complex so we can adopt a medium weight: $1 * 5 = 5$ FPs.

  - The application delivers incoming requests to taxi drivers. This operation is simple so we can adopt a simple weight: $1 * 4 = 4$ FPs.

  - The application provides to taxi drivers a certain amount of information about optimal routes and the ongoing ride. This operation is quite complex so we can adopt complex weight: $1 * 7 = 7$ FPs.

  - The application allows taxi drivers to visualize the dashboard. This operation is not too complex so we can adopt medium weight: $1*5 = 5$ FPs.

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

- **External Inquiry**: The application allows users to access some information. In particular user can see:

  - Information about the system if he is a developer.

  - The summary of receipts if he is a passenger.

  - The summary of rides if he is a taxi driver.

Considering all these 3 operations as simple, 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 SLOC and then use another approach such as COCOMO II to estimate the effort.

We use the standard conversion coefficient to compute the number of Source Lines Of Code (SLOC) starting from the FPs. Therefore:

$SLOC = 46 * FPs = 5658$

where 46 is the value of the coefficient related to the specific framework we are using, Java EE, taken from the table at this web page: http://www.qsm.com/resources/function-point-languages-table

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_modelman 2000.0.pdf.

Following this document we have compiled the two tables below, and then we have used the online calculator at this web page: http://csse.usc.edu/tools/ COCOMOII.php to compute 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

USC CSSE

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

Software Size   Sizing Method Source Lines of Code

	SLOC	% Design Modified	% Code Modified	% Integration Required	Assessment and Assimilation (0% - 8%)	Software Understanding (0% - 50%)	Unfamiliarity (0-1)
New	5658						
Reused		0	0				
Modified							

Software Scale Drivers

Precedentedness	Very Low	Architecture / Risk Resolution	Low	Process Maturity	High
Development Flexibility	Nominal	Team Cohesion	Extra High		

Software Cost Drivers

Product / **Personnel** / **Platform**

Product		Personnel		Platform	
Required Software Reliability	Nominal	Analyst Capability	Low	Time Constraint	Nominal
Data Base Size	Nominal	Programmer Capability	Nominal	Storage Constraint	Nominal
Product Complexity	Nominal	Personnel Continuity	Nominal	Platform Volatility	Nominal
Developed for Reusability	High	Application Experience	Low		
Documentation Match to Lifecycle Needs	Nominal	Platform Experience	Low	**Project**	
		Language and Toolset Experience	Nominal	Use of Software Tools	Nominal
				Multisite Development	Very High
				Required Development Schedule	Nominal

Maintenance Off

Software Labor Rates
Cost per Person-Month (Dollars) 2000
Calculate

Results

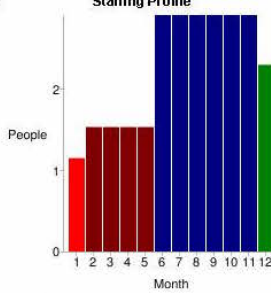**Software Development (Elaboration and Construction)**

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

Created by Ray Madachy at the Naval Postgraduate School. For more information contact him at 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

    - T4.1 : Client Side

    - T4.2 : Server Side

    - T4.3 : Database

- 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 (Figura 4.1) 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).

| ATTIVITA'		PERIODO		
LIVELLO 1	LIVELLO 2	DATA INIZIO	Durata (gg)	DATA FINE
Task 1	Feasibility Study	October 15, 2015	7	October 22, 2015
Task 2	Analysis of Requirements and Specifications	October 15, 2015	22	November 6, 2015
Task 3	System Design	November 7, 2015	27	December 4, 2015
Task 4	Coding and Unit Testing Client	December 5, 2015	162	May 15, 2016
	Coding and Unit Testing Server	December 5, 2015	192	June 14, 2016
	Create and Populate Database	May 16, 2016	29	June 14, 2016
Task 5	Integration Test and System Test	June 15, 2016	40	July 25, 2016
Task 6	Deployment	July 26, 2016	15	August 10, 2016
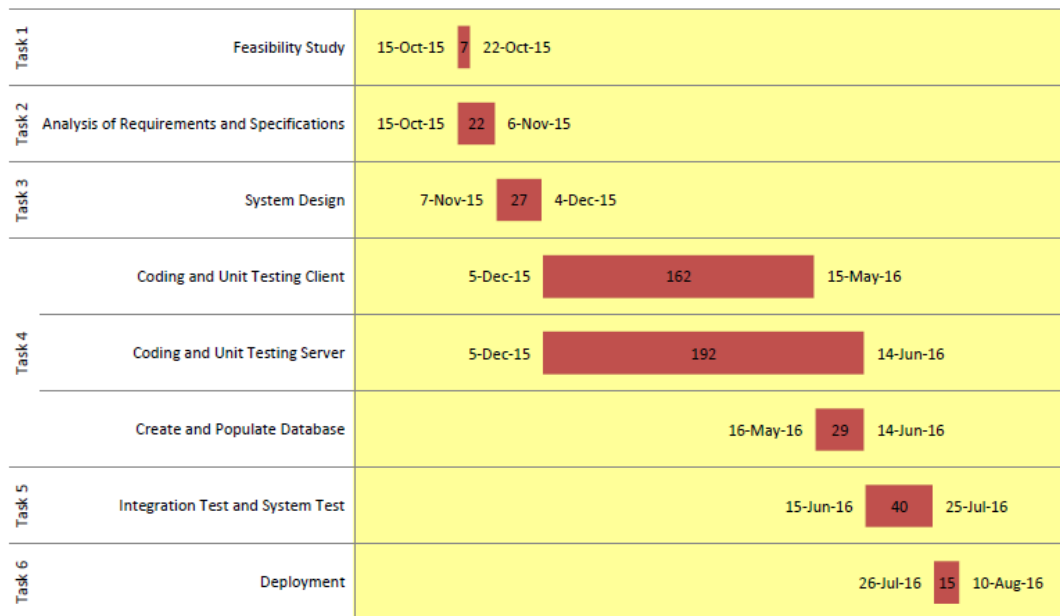
Figure 4.1: Project schedule

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.