Software Engineering 2
“myTaxiService”

Project Plan Version 1.0

2/1/2016

Politecnico di Milano A.A. 2015-2016

Milica Jovanovic (mat. 835953);Pavle Vidanovic (mat. 854472)

Contents

1 Introduction 2

1.1 Revision History 2

1.2 Purpose and Scope 2

1.3 Definitions and Abbreviations 3

1.4 Reference Documents 3

1.5 Document Overview 3

2 Function Points 4

2.1 Brief Introduction 4

2.2 FP Estimation 4

2.2.1 Internal Logic Files 4

2.2.2 External Interface Files 5

2.2.3 External Inputs 5

2.2.4 External Inquiries 6

2.2.5 External Outputs 6

2.2.6 Resuming 6

3 Effort Estimation COCOMO II 7

3.1 Brief Introduction 7

3.2 Scale Drivers 7

3.3 Cost Drivers 8

3.4 Effort Equation 8

3.5 Schedule Estimation 8

4 Task Allocation 9

5 Resources Allocation 9

6 Risk Management 10

7 References 10

1 Introduction

1.1 Revision History

Version	Data	Authors	Summary
0.1	21/01/2016	Pavle Vidanovic Milica Jovanovic	Initial Draft
0.2	23/01/2016	Pavle Vidanovic Milica Jovanovic	Project Plan Version 0.2
1.0	02/01/2016	Pavle Vidanovic Milica Jovanovic	Project Plan Version 1.0

1.2 Purpose and Scope

The purpose of this document is to define plan for testing, integration testing and verifying that system development during the project complies with the requirements of Requirement document and Design Document. This document also presents test results in order to determinate if the application meets predetermined requirements and functionalities.

The aim of this project is to develop and implement myTaxiService, an application similar to Uber, which makes the process of assigning an available taxi vehicle to possible passengers.

The developed system should allow new users to register. Users, once logged in, should be able to:

- request a taxi
- reserve a taxi
- cancel a ride
- check taxi availability around him
- receive a confirmation with information about the assigned vehicle and ETA once taxi is requested
- create/maintain user profile
- report a taxi driver

The developed system should allow new taxi drivers to register. Drivers, once logged in, should be able to:

- inform the system about their availability
- confirm/decline that they are going to take care of a certain call
- create/maintain taxi driver profile
- report a passenger

The system should keep information about new arrived requests, as well as the confirmed rides. A ride should have and id number, information about the passenger that requested the ride, as well as the code of the assigned vehicle and ETA. System should also keep information about taxi queues connected to particular zone of the city and ensure fair management of the queues. Developed system should keep information about the list of reservations made by passengers, such as id number of the reservation, information about the passenger that made the reservation and the time of reservation and time of the ride.

1.3 Definitions and Abbreviations

DB	Data base
FP	Function Points
COCOMO	Constructive Cost Model
PM	Person-Month
UFP	Un-adjusted Function-Points
ILF	Internal Logic File
EIF	External Interface File
EIQ	External Inquiry
EI	External Input
EO	External Output
SLOC	Source Lines of Code
HTML	Hypertext Markup Language
PHP	Hypertext Preprocessor
JS	JavaScript

1.4 Reference Documents

- RASD - RASD myTaxiService - final v2.0
- DD - DD myTaxiService - final
- Assignment 5 – Project Plan
- Example of usage FP and COCOMO for Assignment 5

1.5 Document Overview

The document is essentially structured in six parts:

- Chapter 1: Introduction, basic overview of software to be developed, revision histories and abbreviations and reference documents
- Chapter 2: Functional Points, calculation of the main functional points by categories and SLOC estimation
- Chapter 3: Effort Estimation COCOMO II, definition of Scale and Cost drivers, calculation of effort, schedule and staff number
- Chapter 4: Task Allocation
- Chapter 5: Resources Allocation
- Chapter 6: Risk Management, list of main possible risks and mitigation techniques
- Chapter 7: References

2 Function Points

2.1 Brief Introduction

The Function Point estimation approach is based on the amount of functionalities in software and their complexity. FP estimators are useful since they are based on information that is available early in the project life cycle. To perform this estimation we've based our parameters on the following tables, taken from COCOMO II.

Function Types	Weight		
	Low	Average	High
N.Inputs	3	4	6
N.Outputs	4	5	7
N.Inquiry	3	4	6
N.ILF	7	10	15
N.EIF	5	7	10

Table 1. FP – Weighting Function Points

When we get the total number of FPs for our system, we have to derive the total number of SLOC, by following the shown correlations:

Language	SLOC/FP
JS	47
HTML	34
PHP	67
Total:	**148**

Table 2. Programming Language SLOC correlation

Average SLOC per FP is 49.33, we will round it at 50.

After calculating the FP and derive SLOC we use this data into the COCOMO II effort equation. Following the COCOMO II algorithm at the end we get the PM effort.

2.2 FP Estimation

2.2.1 Internal Logic Files

The myTaxiService system stores information about:

- Users
- Drivers
- Reservations
- Reports
- Requests
- Zones
- Admin
- Taxi vehicles

System stores information about users using myTaxiService system. It stores id, name, surname, phone, email, password, image path. Considering the fields stored about the user we will assume its complexity is ***Average***. Information stored about the driver includes user's fields as well as drive license number and its availability indicator. Taking this into account we will assume that its complexity is as well

Average. Information stored about requests is id, ETA, origin, valid; its complexity is ***Low***. Information stored about reservations is the fields of request object with additional four fields; time of reservation, time of ride, destination and description. We will assume complexity of reservation object is Average. Report object has four fields; id, description, user id and driver id. We will assume it's complexity as Low. Information stored about zone is id, radius, longitude and latitude; its complexity is as well Low. Admin object has two fields, username and password; its complexity is Low. Taxi vehicle has three parameters id, type and number of seats; thus its complexity is Low.

Taking into consideration above description our system handles with three Average and five Low complexity objects.

ILF	Complexity	FP
User	Average	10
Driver	Average	10
Request	Low	7
Reservation	Average	10
Report	Low	7
Taxi Vehicle	Low	7
Zone	Low	7
Admin	Low	7
Total:		**65**

Table 3. ILF function points

2.2.2 External Interface Files

myTaxiService system user three external APIs: GoogleMaps API, GooglePlaces API and Gmail API. Invoking GoogleMaps API happens often, at least once when user requests a taxi, or when checking on the map if there are taxi vehicles available near him. Google Places API is invoked when user request or reserve a taxi; when he specify his origin or destination address. Thus Google Places API is invoked often. Gmail API is invoked only once per user, when he creates account for myTaxiService.

EIF	Complexity	FP
Google Maps API	High	10
Google Places API	Average	7
Gmail	Low	5
Total:		**22**

Table 4. EIF function points

2.2.3 External Inputs

Sign Up and Sign In are simple operations; we will adopt low weight for them. Report a user of the application is as well a simple operation, thus we will adopt low weight for it. Manage user's profile we have considered as average weight operation because the user can change most of his information in his profile such as name, surname, email, phone number etc. Cancel ride is considered to be a low weight operation because user input is a simple click event, the same stands for driver's Set Available operation. Admin's ban User is considered as average weight operation because he has to choose a report from the list of reports and then appropriate user is removed from the system.

Clients	Function	Complexity	FP
Guest	signUp()	Low	3
	signIn()	Low	3
User	report()	Low	3
	manageProfile()	Average	4
	cancelRide()	Low	3

Driver	setAvailable() manageProfile() report() cancelRide()	Low Average Low Low	3 4 3 3
Admin	banUser() signIn()	Average Low	4 3
Total:			**36**

Table 5. EI function points

2.2.4 External Inquiries

Make request inquiry is considered to be high weight operation because it involves user entity, the scheduler entity which has to create the request in the data base and contact the zone manager which is responsible for allocating a taxi vehicle for the requested ride. Similarly the Make reservation operation is considered to be of high weight. Check taxis available and check reservation are considered to be of low weight because they do not involve complex back end logic, they are just querying the data base with appropriate parameters. Confirm or Decline Ride is considered to be of average weight because it involves the Scheduler activity and driver activity and is responsible for notifying the driver about a received request for a ride; thus the back end logic behind this function is quite complex. Check rides is average weight operation taking into consideration the possible size of previously driven rides or current or future rides of a taxi drive; i.e the number of rows in a DB could be high. View reports operation is considered as simple query to the DB which shows the admin user the list of all active reports; i.e this operation is simple so we have accepted low weight for it.

Clients	**Function**	**Complexity**	**FP**
User	makeRequest() makeReservation() checkTaxisAvailable() checkReservation()	High High Low Low	6 6 3 3
Driver	confirmDeclineRide() checkRides()	Average Average	4 4
Admin	viewReports()	Low	3
Total:			**29**

Table 6. EIQ function points

2.2.5 External Outputs

myTaxiService system has three external outputs:

- Notification is sent when user requests a taxi and system finds available taxi in the queue; message is sent to driver
- Acknowledgement is sent back to user when taxi driver accepts ride
- Notification is sent to the user after successful reservation

All three outputs are considered to be high weight operations because they involve interaction of multiple software components of myTaxiService system. Estimation of external output is $3 * 7 = 21\ FP$

2.2.6 Resuming

Taking into consideration of estimation of FP the total estimation is 173 FPs. Obtained value of total FP is used to get estimation of number of lines of code. Using the proposed equation we get 8650 SLOC.

The following table resumes our estimations:

Function Type	Number of FP
ILF	65
EIF	22
EI	36
EIQ	29
EO	21
Total:	**173**

Table 7. Total FP

3 Effort Estimation COCOMO II

3.1 Brief Introduction

The COCOMO II Technique is used to estimate the effort, duration and people needed for the project. All the tables used in this analysis have been taken from COCOMO II, Model Definition; Manual at: http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf

To estimate the effort calculated in *Person/Months* needed, the following formula is used:

$$Effort = 2.94 * EAF * (KSLOC)^{E}$$

E Exponent derived from Scale Drivers

$$E = 0.91 + 0.01 * (PREC + FLEX + RESL + TEAM + PMAT)$$

EAF Effort Adjustment Factor derived from Cost Drivers (product of the effort multipliers corresponding to each of the cost drivers for your project)

3.2 Scale Drivers

Where:

- **PREC** Precedentedness reflects the previous experience of the organization with this type of project. Very low means no previous experience, Extra high means that the organization is completely familiar with this application domain. **PREC** = *nominal*, because the team has previous experience with developing Web applications but in controlled environment such as Bachelor project
- **FLEX** Development Flexibility reflects the degree of flexibility in the development process. Very low means a prescribed process is used; Extra high means that the client only sets general goals. **FLEX** = *nominal*, because there were fixed initial requirements for the project
- **RESL** Risk Resolution reflects the extent of risk analysis carried out. Very low means little analysis, Extra high means a complete a thorough risk analysis. **RESL** = *low*, because the focus of the project was not on risk detection
- **TEAM** Team cohesion reflects how well the development team knows each other and work together. Very low means very difficult interactions, Extra high means an integrated and effective team with no communication problems **TEAM** = *high*, because team members know each other from before
- **PMAT** Process maturity reflects the process maturity of the organization. **PMAT** = *nominal*, because this was not the first time developing such an application

Scale Driver	Factor	Value
PREC	Nominal	3.72
FLEX	Nominal	3.04

RESL	Low	5.65
TEAM	High	2.19
PMAT	Nominal	4.68
Total		**19.28**

Table 8. Scale drivers factors

$$E = 0.91 + 0.01 * 19.28 = 1.1028$$

3.3 Cost Drivers

Scale Driver	Factor	Value
Required Software Reliability	Nominal	1.00
Data Base Size	High	1.14
Product Complexity	High	1.17
Required Reusability	Nominal	1.00
Documentation match to lifecycle needs	Nominal	1.00
Execution Time Constraint	Very Low	n/a
Main Storage Constraint	Very Low	n/a
Platform Volatility	Low	0.87
Analyst Capability	Very High	0.71
Programmer Capability	Very High	0.76
Application Experience	High	0.88
Platform Experience	High	0.91
Language and Tool Experience	High	0.91
Personnel continuity	Very Low	1.29
Usage of Software Tools	Nominal	1.00
Multisite development	Very Low	1.22
Required development schedule	Nominal	1.00
Product:		**0.72**

Table 9. Cost drivers factors

3.4 Effort Equation

This final equation gives us the effort estimation measured in PM

$$Effort = 2.94 * EAF * (KSLOC)^E$$

Where:

$$EAF \rightarrow product\ of\ all\ the\ Cost\ Drivers\ is\ equal\ to\ \mathbf{0.72}$$

$$E \rightarrow exponent\ derived\ from\ Scale\ Drivers\ is\ equal\ to\ \mathbf{1.1028}$$

$$KSLOC \rightarrow estimated\ lines\ of\ code\ using\ the\ FP\ analysis\ is\ equal\ to\ \mathbf{8.6}$$

With these parameters we can compute the Effort value as following:

$$Effort = 2.94 * 0.72 * 8.6^{1.1028} = \mathbf{22.71PM}$$

3.5 Schedule Estimation

As far as the schedule estimation we are going to use the following formula:

$$Duration = 3.67 * Effort^F$$

Where:

$$F = 0.28 + 0.2 * (E - 0.91) = 0.31856$$

Taking into consideration the calculated parameters, the duration of project should be:

$$Duration = 3.67 * 22.71^{0.31856} = \mathbf{9.92\ Months}$$

The duration calculated above is not close to the actual time we had at our disposal for Software Engineering II Project, which was around 3 months. This may be because COCOMO II is an industry oriented method where a lot more time is spent considering security and scalability issues and projects go through many iterations as the team gets more familiar with the domain of the problem they are facing and because the client's requirements may change over time.

As for the required number of people the estimation is:

$$People = \frac{Effort}{Duration} = \mathbf{2.29\ People}$$

This result shows that ideal number for this project would be three persons, but It would be possible for two persons to finish this project with more effort.

4 Task Allocation

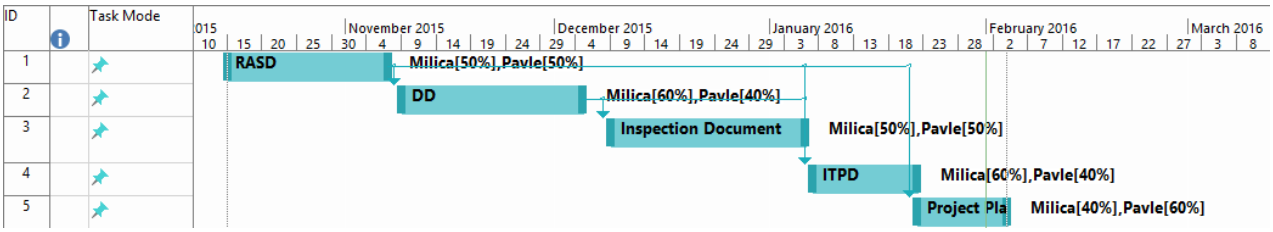

Figure 1. Task allocation Gantt chart

5 Resources Allocation

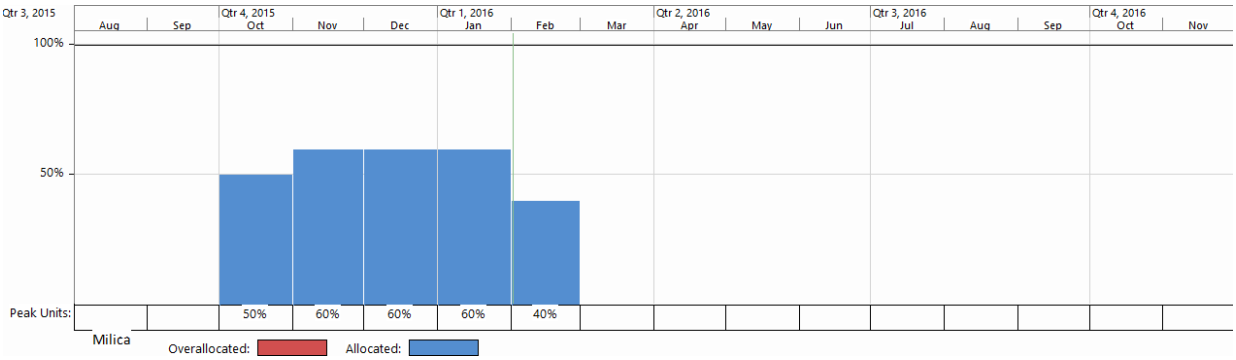

Figure 2. Resource allocation for Milica

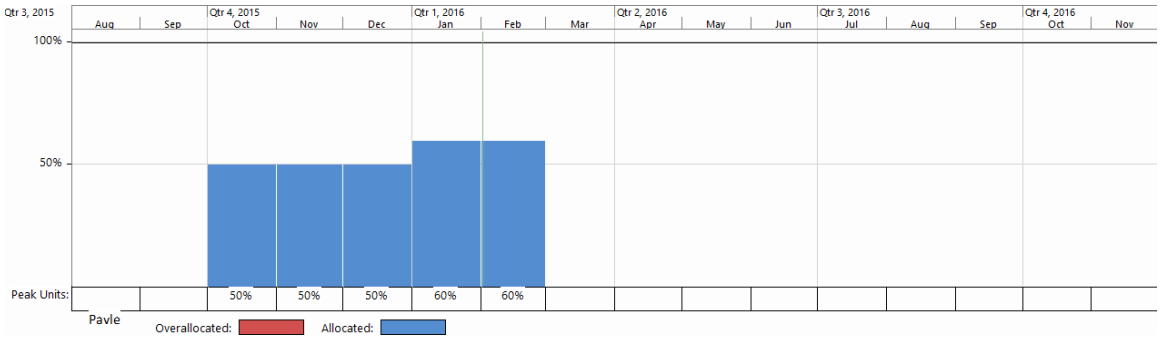

Figure 3. Resource allocation for Pavle

6 Risk Management

Risk Description	Risk Mitigation	Risk Priority
Lack of communication between all the team members	The team should meet at least once a week	High
Underestimate project workload	The team should discuss about project scope in every meeting	Medium
The developed solution doesn't fit the final user's needs	During the requirement specification, team should do some surveys and interviews to find what the users are looking for	High
Application reliability and usability	The team should develop with good practices and develop a list of acceptance tests	Medium

Table 10. Risk management

7 References

- Introduction To Function Point Analysis - http://www.softwaremetrics.com/fpafund.htm
- Overview of COCOMO - http://www.softstarsystems.com/overview.htm
- Risk management - https://en.wikipedia.org/wiki/Risk_management
- ISO/IEC 31010:2009 - Risk Management - Risk Assessment Techniques