



POLITECNICO MILANO 1863

Politecnico di Milano
A.A. 2015/2016
Software Engineering 2
Assignment 5: Project Plan
Version 1.0

Alessandro Baldassari (mat. 841561)
Alberto Bendin (mat. 841734)
Francesco Giarola (mat. 840554)

January 26, 2016

Contents

	Page
1 Introduction	1
1.1 Purpose and Scope	1
1.2 List of Definitions and Abbreviations	1
2 Function Point analysis	2
2.1 Introduction	2
2.2 FP types estimation	2
3 COCOMO II analysis	5
3.1 Introduction	5
3.2 Parameters choice and estimation	5
3.3 Second analysis with automated tool	6
4 Tasks identification and schedule	8
5 Resources allocation	8
6 Risk planning and management	8
7 References	9
8 Appendix	9
8.1 Software and tools used	9
8.2 Hours of work	9

1 Introduction

1.1 Purpose and Scope

The main purpose of the project plan is to plan time, cost and resources adequately to estimate the work needed and to effectively manage risk during project execution. A failure to adequately plan reduces the project's chances of successfully accomplishing its goals.

Project planning generally consists of:

- Identifying deliverables and creating the work breakdown structure;
- Identifying the activities needed to complete those deliverables and networking the activities in their logical sequence;
- Estimating the resource requirements for the activities;
- Estimating time and cost for activities;
- Developing the schedule;
- Developing the budget;
- Resource allocation (organization of work loads);
- Risk planning.

This document is based on an analysis made with two different algorithmic metrics of the system for *myTaxiService*. The first one is the Function Points (FP), which is used to estimate the software dimension (code size), which is directly used to evaluate the cost. The second is the COCOMO II that is used to estimate the efforts required in the development of a project by taking in account: characteristics of people, products and process.

1.2 List of Definitions and Abbreviations

The following acronyms are used in this document:

- FP: Function Points
- COCOMO: COnstructive COst MOdel
- ILF: Internal Logic File
- EIF: External Interface File
- PM: Person-Months
- SLOC: Source Lines of Code
- KSLOC: Thousands of SLOC
- SD: Scale Drivers
- EAF: Effort Adjustment Factor

The following definitions are used in this document:

- Deliverables: are work that are delivered to the customer, e.g. a requirement document for the system.

2 Function Point analysis

2.1 Introduction

The Function Point estimation approach is based on the amount of functionalities in a software and their complexity. Indeed the effort to develop a software project grows with the number of external inputs and outputs, user interactions, files and interfaces used by the system; therefore a weight is associated to all these functionalities and the total effort is computed summing all the partial values.

The parameters used to perform this estimation are summarized in the following tables, taken from COCOMO II, Model Definition Manual at:

http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf.

The schema below defines the weights assigned to every level of complexity for all the FP types.

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

Here is a brief explanation of the FP types:

- Internal Logic 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 from the external environment
- External Output: elementary operation that generates data for the external environment (it usually includes the elaboration of data from logic files)
- External Inquiry: Elementary operation that involves input and output (without significant elaboration of data from logic files)

2.2 FP types estimation

The assignation of the complexity to each functionality is based on the number of fields and interactions with other components required to implement it.

- ILF

Functionalities	Complexity	FP Count
Users	Simple	7
Ride	Complex	15
Request	Medium	10
Zone	Medium	10
Taxi	Simple	7
Payment-Receipts	Simple	7
Total:		56

- EIF

Functionalities	Complexity	FP Count
Payment Data	Medium	7
Google Maps	Medium	7
Push notification metadata	Simple	5
Total:		19

- Ext. INPUT

Functionalities	Complexity	FP Count
Login	Simple	3
SignUp	Simple	3
Create request	Simple	3
Pay	Medium	4
Delete request	Complex	6
Set taxi availability	Simple	6
Accept/reject ride request	Simple	3
Update taxi position	Simple	3
Managers taxi drivers	Medium	4
Create ride	Complex	6
Allocate taxi	Complex	6
Total:		44

- Ext. INQUIRY

Functionalities	Complexity	FP Count
Request history	Simple	3
Manage profile	Simple	3
Ride history	Simple	3
Public API	Medium	4
DB	Medium	4
Total:		17

- Ext. OUTPUT

Functionalities	Complexity	FP Count
SMS	Medium	4
Email	Simple	4
PushNotification	Simple	4
Zone	Simple	10
Total:		17

- Total and conclusions

FP types	Value
ILF	56
EIF	19
Ext. INPUT	44
Ext. INQUIRY	17
Ext. OUTPUT	17
Total:	153

The total number of FPs is then 153.

We can use this number to estimate the number of lines of code needed for the project; to do this a conversion factor is used: 46.

This value is obtained from the average value for J2EE language from the table at <http://www.qsm.com/resources/function-point-languages-table>.

Thus the estimate of SLOC will be:

$$153FPs * 46 = 7038 \text{ SLOC}$$

This result is the starting point for the next evaluation technique, the COCOMO approach.

3 COCOMO II analysis

3.1 Introduction

This estimation is achieved through a complex, statistical model that takes in account the characteristics of the product but also of people and process. The result of this technique is the estimation of Person-Months required to develop the project.

The COCOMO II calculations are based on the estimated of the software dimension in source lines of code (SLOC). Two very important metrics are used for this evaluation:

- Scale Drivers: the 5 SD determine the exponent used in the “effort equation”
- Cost Drivers: the 17 Cost Drivers represent the multiplicative factors that determine the effort required to complete the project

The result, the so-called “*Effort equation*” is:

$$\text{Effort} = 2.94 * \text{EAF} * (\text{KSLOC})^E [\text{Person-Months}]$$

Where:

- *EAF*: Effort Adjustment Factor derived from Cost Drivers (product of the effort multipliers corresponding to each of the cost drivers for the project)
- *E*: Exponent derived from SD
- *KSLOC*: SLOC measured in thousands

Once the effort has been computed it is possible to calculate the number of months required to complete the project with the duration equation:

$$\text{Duration} = 3.67 * (\text{Effort})^E [\text{Months}]$$

Where:

- *Effort*: is the effort computed above
- *E*: is the schedule equation exponent derived from the five Scale Drivers

When both the *Effort* and the *Duration* are available then the number of people required to complete the project is:

$$N_{\text{people}} = \lceil \text{Effort} / \text{Duration} \rceil [\text{People}]$$

3.2 Parameters choice and estimation

A first estimation is carried out considering a project with all nominal Cost Drivers and Scale Drivers: this means an *EAF* of 1.00 and exponent *E* of 1.0997. The effort equation thus becomes:

$$\text{Effort} = 2.94 * 1.0 * (7.038)^{1.0997} = 25.13 \text{ Person-Months}$$

Proceeding with the computation of the schedule equation the formula with the nominal parameter of *E* = 0.3179 becomes:

$$\text{Duration} = 3.67 * (25.13)^{0.3179} = 10.23 \text{ Months}$$

The size of the team is then:

$$N_{\text{people}} = \lceil 25.13 / 10.23 \rceil = \lceil 2.45 \rceil = 3 \text{ People}$$

The analysis gives the following result: it takes about 10 months of 3 people-work to finish this project.


3.3 Second analysis with automated tool

A second analysis is here presented; it has been done with the help of an online tool (<http://csse.usc.edu/tools/COCOMOII.php>), where it is easier and immediate to evaluate the difference in effort and scheduling as parameters of drivers change.

In particular the variations with respect to the nominal case are the following:

- Precedentedness: LOW, because the team has not much experience on large scale Java Enterprise projects with such high quality and reliability requirements
- Architecture/Risk Resolution: HIGH, since the team wants to avoid mistakes that can be raised during the development at a late stage that could involve the feasibility of the project itself; this is achieved through a thorough analysis of the architecture and the possible risks.
- Required Software Reliability: HIGH, since the expected quality of the system is very high, and on it depend a lot of processes (passenger clients, drivers, the public transportation for whole cities...)

Below are the settings of this analysis.



COCOMO II - Constructive Cost Model

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	7038					
Reused		0	0			
Modified						

Software Scale Drivers

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

Software Cost Drivers

Product	Personnel	Platform			
Required Software Reliability	High ▾	Analyst Capability	Nominal ▾	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	Nominal ▾	Application Experience	Nominal ▾		
Documentation Match to Lifecycle Needs	Nominal ▾	Platform Experience	Nominal ▾	Project	
		Language and Toolset Experience	Nominal ▾	Use of Software Tools	Nominal ▾
				Multisite Development	Nominal ▾
				Required Development Schedule	Nominal ▾

Maintenance Off ▾

Software Labor Rates

Cost per Person-Month (Dollars)

Below is the output of the analysis.

Results

Software Development (Elaboration and Construction)

Effort = 27.6 Person-months

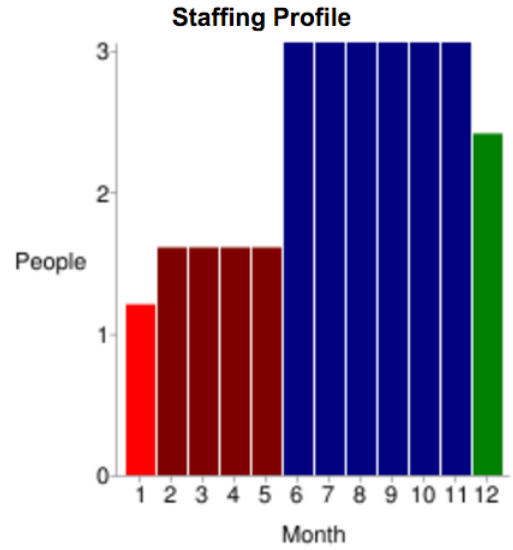
Schedule = 11.0 Months

Cost = \$0

Total Equivalent Size = 7038 SLOC

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	1.7	1.4	1.2	\$0
Elaboration	6.6	4.1	1.6	\$0
Construction	20.9	6.9	3.1	\$0
Transition	3.3	1.4	2.4	\$0



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

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.2	0.8	2.1	0.5
Environment/CM	0.2	0.5	1.0	0.2
Requirements	0.6	1.2	1.7	0.1
Design	0.3	2.4	3.4	0.1
Implementation	0.1	0.9	7.1	0.6
Assessment	0.1	0.7	5.0	0.8
Deployment	0.0	0.2	0.6	1.0

The slight variation due to the different parameters does not really change the estimation:

- Duration: from about 10 to 11 months.
- Size of the team: always 3 people required ($N_{people} = \lceil 27.6/11 \rceil = \lceil 2.5 \rceil = 3 \text{ People}$)

This obvious result is a direct consequence of the increase in risk control and high-quality-software objectives prefixed with the parameters.

4 Tasks identification and schedule

5 Resources allocation

6 Risk planning and management

7 References

Material from Wikipedia

- Project management: https://en.wikipedia.org/wiki/Project_management#Planning

8 Appendix

8.1 Software and tools used

- TeXstudio 2.10.6 (<http://www.texstudio.org/>) to redact and format this document.
- Astah Professional 7.0 (<http://astah.net/editions/professional>)

8.2 Hours of work

The time spent to redact this document:

- Baldassari Alessandro: 12 hours.
- Bendin Alberto: 12 hours.
- Giarola Francesco: 12 hours.