Project Plan

Figure 1: Politecnico di Milano

Version 1.0

Release date : 02/02/2016

- Claudio Cardinale (mat. 849760)
- Gilles Dejaegere (mat. 853950)
- Massimo Dragano (mat. 775392)

Contents

1. Introduction

1.1. Revision history

Date	Version	Description	Authors
02/02/2016	1	Original Version	C. Cardinale, G. Dejaegere and M. Dragano

1.2. Purpose and Scope

1.3. List of Definitions and abbreviations

- Function Points estimation : the function point estimation is an estimation of the size of an specific software in terms of line of codes.
- SLOC: **TODO**
- FP: **TODO**

1.4. List of Reference Documents

2. Estimate size, effort and cost

2.1. Function points

Function Points estimation : the function point estimation is an estimation of the size of an specific software to be based on its functional requirements. The count of function points of the software is language and technology independant. The size in terms of lines of codes of this software can be calculated using the following formula :

LOC = FP * AVC

where : * LOC = lines of code * FP = total estimated function points of the software * AVC = language specific constant varying from 2-40 for a fourth generation programming language.

Function points are classified under different types. For each type, a different weight is also assigned for each function point according if the concerned function is estimated as being simple, complex, or average. For the estimation of the size of our MyTaxiService application, we will uses the weights indicated in the table here under. These numbers are taken from the slides "Cost Estimation" provided by the professor Damian Andrew Tamburri.

function type	**Low**	**Average**	**High**
Internal Logic Files	7	10	15
External Interface Files	5	7	10
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6

2.1.1 Internal Logic Files

The different ILFs that will be used by our systems can be deduced from the class diagram representing the `model` of our application that is located in the RASD and is shown here under.

[Model taken from the RASD][model]

For each component, its complexity and the associated function points are listed in the table here under :

ILF	**Complexity = Function Points**
Zone	High = 15
Driver and Taxi	High = 15

ILF	Complexity = Function Points
Request	Average = 10
Ride	High = 15
StopPositions and Client	Low = 7
Position	Low = 7

Since each Driver has exactly one taxi, and each stop position has exactly one client, these entities are managed and stored together for more simplicity.

2.1.2 External Interface Files

2.1.3 External Inputs

2.1.4 External Outputs

2.1.5 External Inquiries

2.2. COCOMO

2.2.1. Introduction to COCOMO

To proceed with the calculation of SLOC we have to convert FP to it. To do that we need a conversation factor, we have found the following site that gives some converstion factors for some language: http://www.qsm.com/resources/function-point-languages-table but there is not php with laravel, so we considered the factor of J2EE that is analogous with php + laravel since they are booth two MVC web application framework and object languages. So the factor that we will use is **46**.

To perform the estimation we will use the parameters of the offcial table http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf **TO IMPROVE**

2.2.2. Sale driver

Copy table from officialfile

2.2.3. Cost driver

2.2.4. Effort equation

Chose batter names

3. Tasks

Use a table like project manager slides

4. Allocate resources

Use a table like project manager slides

5. Risks

6. Used tools

- Github: for version controller
- Gedit and ReText: to write MarkDown with spell check

7. Hours of work

Claudio Cardinale

Gilles Dejaegere

Massimo Dragano