

Politecnico di Milano

A.A. 2015-2016

Software Engineering 2: “myTaxiService”

Project Plan

Luiza Bentivoglio, Michele Cantarutti

2 February 2016

Summary

**1. INTRODUCTION ............................................................................................................4**

**2. PROJECT ESTIMATION..............................................................................................6**

2.1 SIZE ESTIMATION..............................................................................................................................6

2.2 EFFORT AND COST ESTIMATION………..............................................................................................6

**3. TASKS………………………..............................................................................................6**

**1. Introduction**

This document is meant to be a project plan for the myTaxiService project. The first part of the document will focus on the estimation of the project size (by applying Function Points) and of the effort and cost (by applying COCOMO). Next, we’ll identify the tasks of our project and their schedule, while allocating the resources to each one of them. Lastly, we’ll define the risks of the project.

**2. Project estimation**

**2.1 SIZE ESTIMATION**

To estimate the size of our software, we’re going to use the Function Points approach. Function points measure software size and, by definition, a function point is a unit of measurement to express the amount of business functionality a software provides to a user. We retrieved the functionalities our software must provide from our RASD and then we evaluated the complexity of each one of them. Moreover, we distinguished five categories of functionalities, sticking with the standard FP approach, as in the model laid by Allan Albrecht, as follows:

-**External Input**: elementary operation to elaborate data coming form

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).

-**Internal Logical File** (**ILF**): homogeneous set of data used and managed by the application;

-**External Interface File** (**EIF**): homogeneous set of data used by the application but generated and maintained by other applications;

As for the points given to each functionality, we’re once again sticking with the original model of the FP approach, as shown in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| Function types | Complexity | | |
|  | Simple | Medium | Complex |
| External Inputs | 3 | 4 | 6 |
| External Outputs | 4 | 5 | 7 |
| External Inquiry | 3 | 4 | 6 |
| ILF | 7 | 10 | 15 |
| EIF | 5 | 7 | 10 |

Now that we have defined the estimation model we intend to use and that we have specified what categories of functionalities our system provides, we can pass on to the proper estimation analysis. To do so, we went through each category, and for each category we analyzed the functionalities that belong to it, by rating the complexity of each one of them. Thus, we assigned the amount of points we felt appropriate for each functionality, based on its complexity.

**-Internal Logical File** (**ILF**): the application stores information about users, taxis, queues, zones and users’ requests. All of these entities have a simple structure composed of a small number of fields, apart from the users’ requests which have quite a few fields. Therefore we think it should be necessary to adopt medium complexity for the requests, whereas the other entities should have a simple complexity. We get 4 x 7 + 1 x 10 points, which add up to 38 FPs, concerning ILFs.

**-External Interface File** (**EIF**): there is only one operation that belongs to this category, and that is the payment, which will involve an interaction with an external system, that is a bank. The fact that interactions with an external system can be difficult to manage, coupled with the fact that transferring money requires high security, leads us to deem this operation a very complex one. Therefore we get 10 FPs.

-**External Input**: the application interacts with the users as follows:

• Login/logout: these are simple operations, so we can adopt the simple

weight for them. 2 x 3 = 6 FPs

• Sign up: this is a simple operation, so we can adopt the simple weight. 1 x 3 = 3 FPs

• Making a request/reservation: these operations involve quite a few entities and imply the creation of a new one, therefore, they’re highly complex: 2 x 6 = 12 FPs

**2.2 EFFORT AND COST ESTIMATION**

**3. Tasks**

1.1 TASK

Documentazione:

RASD:

T1: introduction

T2: ACTORS IDENTIFYING

T3: REQUIREMENTS

T4: SCENARIOS IDENTIFYING

T5: UML MODELS

T6: ALLOY MODELLING

T7: USED TOOLS

T1+T2+T3 1 GIORNO

T4+T5 3 GIORNI

T6+T7 2 GIORNI **totale 5 giorni rasd**

DD:

T8: INTROCTION

T9: ARCHITECTURAL DESIGN

T10: ALGORITHM DESIGN

T11: USER INTERFACE DESIGN

T12: REQUIREMENTS TRACEABILITY

**TOTALE: 3 GIORNI**

TEST PLAN:

T13: INTRODUCTION

T14: INTEGRATION STRATEGY

T15: INDIVIDUAL STEPS AND TEST DESCRIPTION

T16: TOOLS AND TEST EQUIPMENT REQUIRED

T17: PROGRAM STUBS AND TEST DATA REQUIRED

**TOTALE: 1 GIORNO**

Programma:

T18: convertire la distribuzione dei component del Sistema nel diagramma delle classi (MVC -> model) **1 giorno**

T19: sviluppare il diagramma delle classi che si occuperà delle informazioni di comunicazione attraverso la rete tra i vari partecipanti del sistema (RMI, socket... ->controller) **3 giorni**

T20: sviluppare il modello + test **3 giorni**

T21:sviluppare il controller+test **4 giorni**

T23: eseguire test della comunicazione tra model+controller **1 giorno**

T24: sviluppo database (registrazione server + modello ) **2 giorni**

T25:svilluppare le varie view : piattaforma web e android (grafica e sviluppo) **2 settimane**

T26:eseguire test finali (view+model+controller) **3 giorni**

Finish

**Tempo totale -> 40 GIORNI.**