

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: COCOMO II**

**2.2.1 Brief Introduction**

This estimation is achieved through a complex, nonlinear model that takes in account the

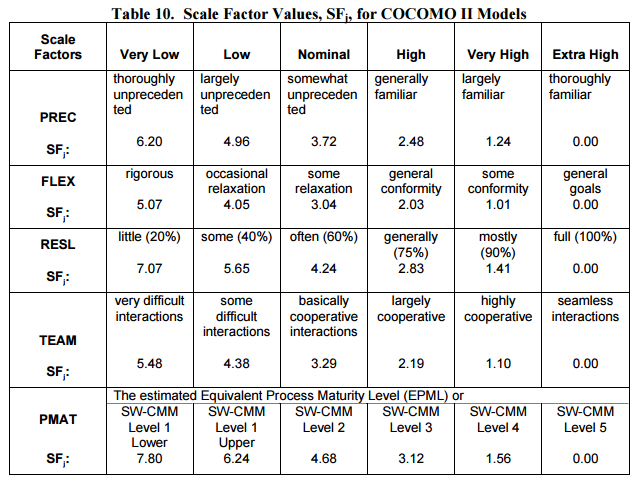
characteristics of the product but also of people and process.

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>

**2.2.2 Scale Drivers**

****

These values are evaluated according to the following table:

• **Precedentedness:**

It reflects the previous experience that we had with this kind of projects. Since for

us this was the first experience using this framework and these developments

methodologies, this value will be low.

• **Development flexibility:**

It reflects the degree of flexibility in the development process. The customer set

the general specifications without going too much in detail, for this reason this

value will be high.

• **Risk resolution:**

Reflects the extent of risk analysis carried out. Thanks to filters, and security

access mostly of the risks were eliminated then this value will be very high.

• **Team cohesion:**

Reflects how well the development team know each other and work together.

*Qui mettiamo che è il primo lavoro che I membri del gruppo fanno insieme, ma dato le competenze organizzative molto simili si può concludere che il livello di coesione tra I membri risulta esser largamente cooperativa, then value will be high.*

• **Process maturity:**

This was evaluated around the 18 Key Process Area (KPAs) in the SEI Capability

Model. Because of the goals were consistently achieved these values will be set to

high, level 3.

The results are resumed in the following table:

|  |  |  |
| --- | --- | --- |
| **Scale Driver** | **Factor** | **Value** |
| Precedentedness Low  Development Flexibility  Risk Resolution  Team Cohesion  Process Maturity | Low  High  Very High  High  High | 4.96  2.03  1.41  2.19  3.12 |

**Total:** 13.71

**2.2.2 Cost Drivers**

**• Required Software Reliability:**

This is the measure of the extent to which the software must perform its intended function over a period of time.

Software failures don't have critical consequences so this parameter is setted to low.

**• Data Base Size:**

*This measure attempts to capture the affect large data requirements have on product development. It translates the effects that large data have in our application.*

*Our test database size is equal to 323.0 KB and the program size dovrebbe venire all’incirca 10000 SLOC, the division D/P = 30.96 and then this parameter has a nominal value.*

**• Product Complexity:**

Set to high according to the new COCOMO II CPLEX rating scale.

• **Required Reusability:**

This cost driver accounts for the additional effort needed to construct components intended for reuse on the current or future projects. This effort is consumed with creating more generic design of software, more elaborate documentation, and more extensive testing to ensure components are ready for use in other applications. In our project there are different reusable components since our aim was to design the system as modular as possible. This parameter is therefore, set to high.

**• Documentation match to life-cycle needs:**

This parameter describes the relation between the provided documentation and the application requirements. Its suitability is setted to nominal since each aspect of our system to be described has been expressed in the RASD or in the DD.

On the other hand, there is no part of those document unrelated to the actual phase of the development the document is addressed to.

**• Execution Time Constraint:**

In our case this parameter is not relevant so is reasonable to set it as very low.

**• Main Storage Constraint:**

This rating represents the degree of main storage constraint imposed on a software system or subsystem. In our project this parameter is not relevant so is setted as very low.

**• Platform Volatility:**

"Platform" is used here to mean the complex of hardware and software (OS, DBMS, etc.) the software product calls on to perform its tasks. If the software to be developed is an operating system then the platform is the computer hardware. If a database management system is to be developed then the platform is the hardware and the operating system. If a network text browser is to be developed then the platform is the network, computer hardware, the operating system, and the distributed information repositories. The platform includes any compilers or assemblers supporting the development of the software system. This rating ranges from low, where there is a major change every 12 months, to very high, where there is a major change every two weeks.

*In our application is reasonable to consider as platforms the DBMS, the operating*

*system, the browser that perform injections and the hardware as far as the*

*environment concerns. We have to consider also the compiler and the webServer*

*that has taken an important role in the developing phase. (e l applicazione mobile come piattaforma…) The platform shouldn't*

*change too often so this value is set to low.*

**• Analyst Capability:**

Design and analysis abilities should be set to high, since we intentionally

dedicated a lot of effort in analysing the problem requirements and its potential

integration in a real word scenario. In particular, not only we can grant that the

requirements have been correctly studied and accomplished, but also that our

design makes our application actually useful for an end user, providing each of the

basic functionalities he may need. In particular we resolved any ambiguity present

in the initial description and explained our solution in the RASD.

**• Programmer Capability:**

This parameter is evaluated according to our degree of cooperation, due to some

small problems on it, this value it's set to high.

**• Application Experience:**

Our project experience is evaluated according to our previous experience in web

projects and also according to our abilities in programming in Java and most

importantly in the Java EE framework. Since this is our first experience in this

typology of project this value is equal to low.

**• Platform Experience:**

Our average knowledges about platforms as: databases, user interfaces and serverside

development are around 1 year, so this parameter is setted as nominal.

**• Language and Tool Experience:**

This parameter reflects the same experience of the previous one, to it's setted to

nominal too.

**• Personnel continuity:**

This parameter is relevant in particular since in the current case our available time

is less than half a year. For this reason we set it to very low.

**• Usage of Software Tools:**

We used NetBeans with Maven to manage dependencies of our project as libraries

and development kits and Git for the repository management. The most

appropriate value is nominal.

**• Multisite development:**

This parameter reflects how we handled the distribution of development over

distance and multiple platforms. We've used phones,mail and Skype also with

screen sharing, so this value is setted to extra high

**• Required development schedule:**

Our efforts were well distributed over the available development time, but

regardless of this fact, the implementation required high efforts at the later phases.

Mainly this is due to the fact that we expanded the initial problem description in

the more complex and profitable way for a real world application. For these reason

this parameter should be set to high.

Our results are expressed in the following table:

|  |  |  |
| --- | --- | --- |
| **Scale Driver** | **Factor** | **Value** |
| Required Software Reliability  Data Base Size  Product Complexity  Required Reusability  Documentation match to life-cycle needs  Execution Time Constraint  Main Storage Constraint  Platform Volatility  Analyst Capability  Programmer Capability  Application Experience  Platform Experience  Language and Tool Experience  Personnel continuity  Usage of Software Tools  Multisite development  Required development schedule |  |  |

**Product:**

**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.**