Politecnico di Milano
Software engineering 2

myTaxiService

Project Plan (PP)

Version 1.0
Release date 2nd february 2016

790021 Antenucci Sebastiano
852461 Buonagurio Ilaria

Prof.ssa Di Nitto Elisabetta

Table of contents

1 Introduction

This document provides the evaluation of time and resources necessary to the development of myTaxiService software. It computes them with FPA and COCOMO II. Furthermore, it analyses in detail the tasks to be computed, the resources to be allocated, the project schedule and analyzes the risks linked to this project.

2 Project size

In this section we will estimate the project size using the Function Point analysis.

2.1 Function Point Analysis

In order to estimate the project's size we used the Function Points Analysis (FPA). It is a method based on software functionalities and their complexity analysis that allows us to estimate the project size; it is assumed that the number of different data structures is a good size indicator. We used this approach because it only requires the software specification that is completely available before the project begins.
FPA uses five entities that are the following:

- Number of Internal Logic Files (ILF) : this concerns data generated by the system, and used and maintained by the system, such as, for example, an index file.

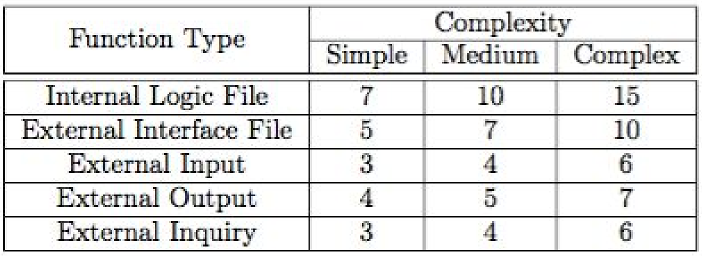
- Number of interfaces (External Logic Files ELF): this concerns data that is output to another application, or is shared with some other application.

- Number of inquiry types (External Inquiry EQ): inqury types concern input that controls the execution of the program and does not change internal data structures. Example of inquiry types are: menu selection and query criteria.

- Number of input types (External Input EI): the input types refer only to user input that result in changes in data structures. It does not concern user input which is solely concerned with controlling the program's execution. Each input type that has a different format, or is treated differently is counted.

- Number of output types (External Output EO): it follows the same counting scheme used for EI.

A further refinement is provided by applying complexity to data types. Each input type has a number of data element types (attributes), and refers to zero or more other file types. We can retrieve complexity division and the relative weights at http://csse.usc.edu/csse/ in particular in chapter 2.2. We report the weights table and remind to the document mentioned before for the complexity division table.



Function Type	Complexity		
	Simple	Medium	Complex
Internal Logic File	7	10	15
External Interface File	5	7	10
External Input	3	4	6
External Output	4	5	7
External Inquiry	3	4	6

2.2 Estimation

In this paragraph we will provide an estimation of the myTaxiService project based on the Function Point Analysis, intended as described in 2.1, and the RASD and DD documents provided with this one.

We can compute the total amount of FPs related to this project by summing all the FPs of the five categories.
In particular, the total amount of FPs related to our project is:

TOT = 49 + 20 + 56 + 14 + 18 = 156 FPs.

We can use the value of TOT in order to retrieve an esteem of the Kilo Lines Of Code (KLOC) of our project and then use another approach, such as COCOMO, to estimate the effort, or we can use it directly to estimate the effort, but this approach need some previous projects to be compared to this one.
We use the first approach and in particular we can retrieve the conversion factor from the following document:
http://www.qsm.com/resources/function-point-languages-table
since in the original reference document there is no value for JEE.
We take the average value, that is 46, so the lines of code are:

LOC = TOT x 46 = 156 x 46 = 7176 LOC

7176 LOC = 7,176 KLOC

We can use an approximation of 8 KLOC since in the FP model functionalities are not fully described by classification, and the model does not capture completely the complexity of the interaction with external services.

2.2.1 Internal Logic Files

The application needs to store data about:

- Taxi driver: is considered as medium complexity since it has the attributes username, password and telephone stored as strings and it is in a 1:1 relationship with a taxi and in a 1:N relationship with a call.

- Taxi: is considered as simple since it has an identificator and a plate both as strings, it is in a 1:1 relationship with a taxi driver.

- Call: is considered as complex because, even if it has only one attribute it is extended by quick call and reservation, that have to make a lot of actions and have a lot of attributes: departure time, departure location, departure date, arrival location. Moreover call is in a 1:1 relationship with taxi driver and user.

- User: is considered as medium complexity since it has the attributes name, surname, telephone, email, password all saved as a string and it is in a 1:1 relationship with a credit card and in a 1:N relationship with a call.

- Credit card: is considered as simple since it has number, type, CCV and expire date all saved as strings and it is in a 1:1 relationship with user.

So the total number of FPs for the ILFs is:

10 + 7 + 15 + 10 + 7 = 49 FPs.

2.2.2 External Logic Files

The application has to interact with: a map service in order to retrieve information about taxi's location; a payment service in order to manage payments and billings.
Both these interactions are considered as complex because there are a lot of instances involved and the result format is not easy to handle.

So the total number of FPs for the ELFs is:

10 + 10 = 20 FPs.

2.2.3 External Inputs

The application allows the user to log in/ log out and sign up that are simple operations. Another simple operation is the one that allows user to manage its profile. It is simple as it involves only one data type.
The application also permits to the user to make a reservation or a quick call and to delete the last reservation and quick call. This four operation can be considered of average complexity since they involve two file types.
As for the taxi driver, it is allowed to log in, log out, manage his profile , accept or discard a call and change his availability. All this operations are considered as simple since they involve one or two data types.

So the total number of FPs for the EIs is:

3 + 3 + 3 + 3 + 4 + 4 + 4 + 4 + 3 + 3 + 3 + 3 + 3 + 3 = 56 FPs.

2.2.4 External Output

The application sends notifications to both users and drivers. It involves more that three data types and seven data elements so they can be considered of high complexity.

So the total number of FPs for the EOs is:

7 + 7 = 14 FPs.

2.2.5 External Inquiry

The application allows users to request information about their profile, the notifications they receive, and their ride history. Drivers, instead, can request information about their calls, their notification and their profile. All this six queries can be considered as simple since they involve one or two data types and from five to fifteen data elements.

So the total number of FPs for the EQs is:

3 + 3 + 3 + 3 + 3 + 3 = 18 FPs.

3 Effort and cost estimation

In this section we will deal with the effort and cost estimation computed with COnstructive COst MOdel II (COCOMO II).

3.1 COCOMO II

It can be found at http://csse.usc.edu/tools/COCOMOII.php. We consider our project with only nominal software scale drivers and software cost drivers product. So the effort adjustment factor is EAF = 1.0; the exponent derived from scale drivers of E =1.0997.
In order to compute the effort we need to use the formula:

$$\text{effort} = 2.94 \times \text{EAF} \times \text{KLOC}^{E}$$

In our case we have:

$$\text{effort} = 2.94 \times 1.0 \times 7.176^{1.0997} = 25.678 \text{ Person/Month}$$

Now we calculate the duration of the project with the formula:

$$\text{duration} = 3.67 \times \text{effort}^{E}$$

In this case we consider the exponent E = 0.3179.

So we have:

$$\text{duration} = 3.67 \times 25.678^{0.3179} = 10.298 \text{ Months}$$

In order to compute the cardinality of our work team we can use the formula:

$$\text{number of people} = \text{effort} / \text{duration}$$

that in our case is:

$$\text{number of people} = 25.678 / 10.298 = 2.49 \text{ People}$$

We can use two people.

4 Tasks & scheduling

In this section we will analyze the tasks that need to be accomplished in order to build the software.

First of all we need to build the documentation necessary to the implementation of the project in this order:

- The Requirements Analysis and Specification Document (RASD)
- The Project Plan (PP)
- The Design Document (DD)
- The Integration Test Plan (ITPD)

After that we can begin the implementation of the project and the testing as stated in the ITPD.

For the creation of the RASD all the members of the group are involved. When it is delivered one member can produce the PP and then help on the DD, and another member can focus on the DD entirely.
After that, one member produces the ITPD while the other begins the implementation of the project following the testing as stated in the ITPD.

We plan to finish the project by the end of january 2016, starting on october 2015.

5 Risks

In this section we will define risks, we will compute their relevance and we will provide some recovery actions to be performed when the risk occurs.

The first we will take in consideration is linked to technical risks and in particular with staff size and experience: since there is no expert project leader and this is the first project for all the memebers of the team there is a risk linked to the implementation of the project and quality of the code. This is quite a relevant risk, since if we can’t run our application we can not deliver it. In order to avoid this risk all group members will study the programming language and how to build the correct documentation for the project. Both the members of the group will actively participate to lectures and, if something is not clear will use the class forum or contact the professor or her assistants.

There could also be some business risks, but we are not going to sell our application so we don't take them in consideration.