

POLITECNICO DI MILANO

A.A. 2015-2016

SOFTWARE ENGINEERING 2: “myTaxiService”

Project Plan Document

Andrijana Mirchevska (838622)

Marija Mavcheva(838647)

22 January 2016

# Introduction

In this document we will evaluate the time and the resources that are necessary for the development of the MyTaxiService application.

In order to do the analyses for the project, the Function Points technique is used and the results have been compared with the dimension of the project.

Also COCOMO approach is used to evaluate the effort for the implementation of MyTaxiService application, and the results have been compared with the time actually spent.

# Function Points Approach

The Function Points is a technique that allows to evaluate the effort needed for the design and implementation of a project. We decided to use this approach also for our project, in order to evaluate the application dimension that is based on the functionalities of the application.

The list of functionalities is obtained from the RASD, and they`re grouped into:

***Internal Logic File (ILF)*** represents a homogeneous set of data used and managed by the application. In MyTaxiService application this corresponds to the Database table.

***External Interface File (EIF)*** represents a homogeneous set of data used by the application but generated and maintained by other applications. In our case this corresponds to the two external platforms: Google Maps API and Mailing Services API.

***External Input*** is an elementary operation to elaborate data coming from the external environment.

***External Output*** is an elementary operation that generates data for the external environment.

***External Inquiry*** is an elementary operation that involves input and output operations.

Every FPA (Function Point Analysis) must begin with grouping the components of the system we are analyzing. That's why the five groups (listed above) are distinguished. Once the components are selected and grouped, we can turn to analyzing itself.

Basically people solve problems by dividing them into smaller parts. And so is with measuring the application complexity with Function Points Method. So, instead of trying to evaluate the application as a whole, we need to rate each of the selected groups.

We need to classify the complexity of each category. We therefore have three possibilities - the complexity could be simple, medium, or complex. Then, the thing is to count the scores following the rules tabled below:

|  |  |  |  |
| --- | --- | --- | --- |
| Function Type | Complexity | | |
| Simple | Medium | Complex |
| Internal Logic File (ILF) | 7 | 10 | 15 |
| External Interface File (EIF) | 5 | 7 | 10 |
| External Input | 3 | 4 | 6 |
| External Output | 4 | 5 | 7 |
| External Inquiry | 3 | 4 | 6 |

So the number of Function Points is computed as the weighted sum of function types using the coefficients of the table above. We perform the calculations step by step:

* **Internal Logic Files (ILFs):**MyTaxiService application includes a number of ILFs that will be used to store the information about users, drivers, requests, reservations, reports, and notifications.

So the internal logic files users, drivers, reports and notifications have simple structure as it is composed of a small number of files, thus they adopt simple weight. And the number of function points is: 4 x 7 = 28 FPs.

The requests and reservations internal logic files have a more complex structure, which means they adopt medium weight. The number of function points is: 2 x 10 = 20 FPs.

**The total FPs concerning ILFs is 28 + 20 = 48 FPs.**

* **External Interface Files (EIFs):**Our application has two EIFs, and those are the APIs for Google Maps and Mailing Services.

Google Maps external interface file has a complex structure and it adopts a complex weight. Its number of function points is: 1 x 10 = 10 FPs.

The Mailing Service external interface file adopts a simple weight. Thus its number of function points is: 1 x 5 = 5 FPs.

**The total number of FPs concerning EIFs is 10 + 5 = 15 FPs.**

* **External Inputs:**The application interacts with the user to allow him/her to:
* *Register*: this is a simple operation and it adopts the simple weight: 1 x 3 = 3 FPs.
* *Login/Log out****:*** these are simple operations and they adopt the simple weight: 2 x 3 = 6 FPs.
* *Change email, change password, change avatar:* these are simple operations and they adopt the simple weight: 3 x 3 = 9 FPs.
* *Request a taxi/make reservation:* these are complex operations since they include interaction among the following components: user, request/reservation, taxi driver, and google maps API. And they adopt the complex weight: 2 x 6 = 12 FPs.
* *Cancel reservation:* this operation involves interaction among the following components: user, reservation, taxi driver, and google maps API. But still is of medium structure so it adopts the medium weight: 1 x 4 = 4 FPs.
* *Report user:* this operation involves interaction among the following components: user, administrator, and taxi driver. It is of medium structure so it adopts the medium weight: 1 x 4 = 4 FPs.
* *Confirm/decline a taxi call:* these operations involve interaction among the following components: taxi driver, request/reservation, and user. It is of medium structure so it adopts the medium weight: 2 x 4 =8 FPs.
* *Managing requests/reservations:* this is a simple operation and it adopts the simple weight: 2 x 3 = 6 FPs.
* *Ban user:* this operation involves interaction among the following components: user, administrator, and taxi driver. It is of medium structure so it adopts the medium weight: 1 x 4 = 4 FPs.

**The total number of FPs concerning EI is 3 + 6 + 9 + 12 + 4 + 4 + 8 + 6 + 4 = 56.**

* **External Outputs:**
* After making a request/reservation the application will send to the user a notification in form of confirmation for the ride. It includes interaction among user, request/reservation, and taxi driver. So it is of complex structure and it adopts complex cost: 1 x 7 = 7 FPs.
* After making a reservation, the application will notify the user at least 2 hours before the ride takes place (since the user must make the reservation at least 2 hours before the ride). It includes interaction among user, request/reservation, and taxi driver. So it is of complex structure and it adopts complex cost: 1 x 7 = 7 FPs.

**The total number of FPs concerning EO is 7 + 7 = 14.**

* **External Inquiries:** None
* **Total FP number and summary**: We have computed the following value for the unadjusted FPs: 48 + 15 + 56 + 14 + 0 = 133. This value can be used to estimate the effort in case we have some historical data that tell us how much time we usually take for developing a FP. Otherwise, it can be used as a basis to estimate the size of the project in KLOC and then use another approach such as COCOMO to estimate the effort.

# COCOMO Approach

To calculate the average SLOC by using the FPs that are calculated above, we’ve taken the average conversion factor given at http://www.qsm.com/resources/function-point-languages-table, which is 46. This is an updated version that adds J2EE of the table included in official manual for COCOMOII.

*133 FPs \* 46 = 6 118 SLOC*

We consider the project with all “Nominal” (i.e., normal) Cost Drivers and Scale Drivers would have an EAF of 1.00 and exponent, E, of 1.0997. So we calculate the effort with the following formula:

effort = 2.94 \* EAF \* (KSLOC)E

and we obtain:

effort = 2.94 \* (1.0) \* (6.118)1.0997 = 21.5 Person/Months

Assuming that the project is expected to consist of *6 118* source lines of code, COCOMO II estimates that 21.5 Person/Months of effort is required to complete it.

**EAF** - Effort Adjustment Factor derived from Cost Drivers

**E** - Exponent derived from Scale Drivers

Now we calculate the duration (schedule) of the project in month, which is based on the effort predicted by the effort equation:

Duration = 3.67 \* (effort)SE

We consider the exponent SE of 0.3179 (calculated from new scale drivers for schedule) and we obtain:

Duration = 3.67 \* (21.5)0.3179 = 9.73 Months

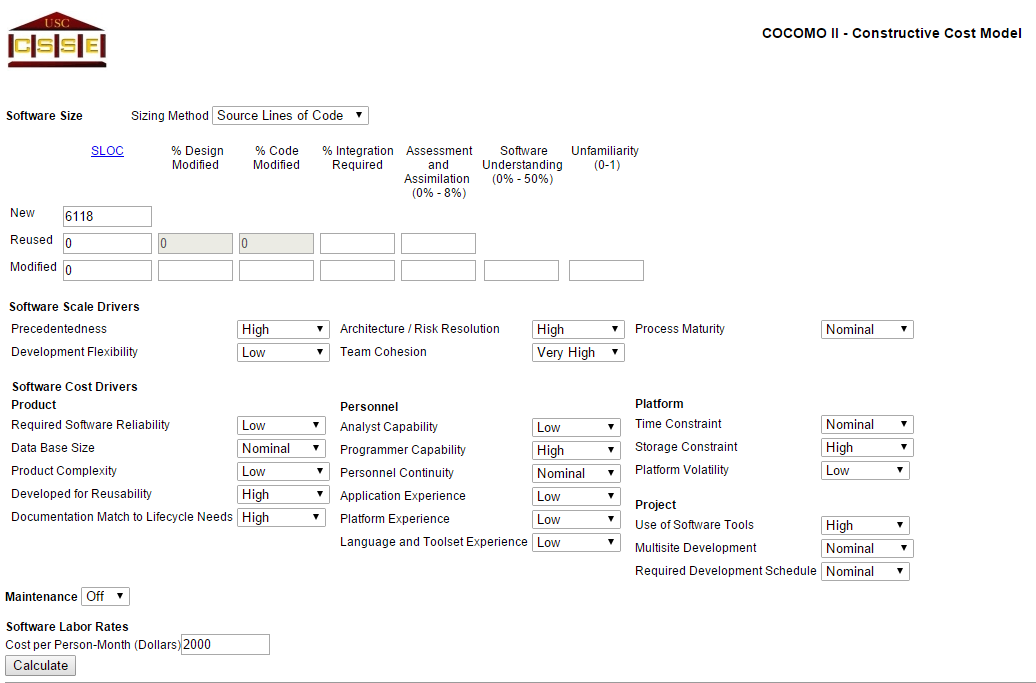
Also the number of people needed to complete the project is estimated. The following formula is used:

Npeople = Effort / Duration

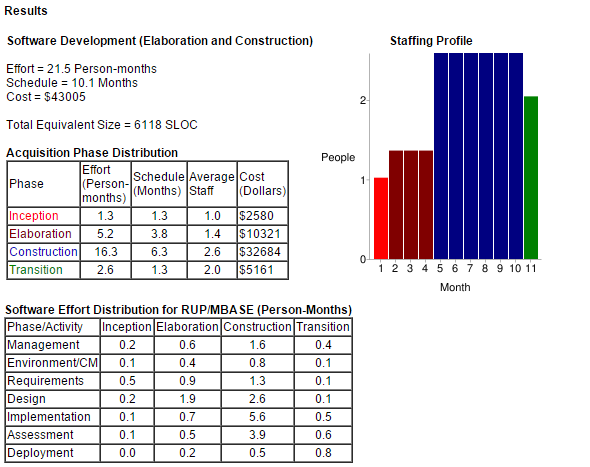
So we obtain:

Npeople = 21.5 / 9.73 = 2.21 ~ 2 people

In order to obtain more precise results for the COCOMO II model we used the online tool (<http://csse.usc.edu/tools/COCOMOII.php>). Below is the report of the site and the choice made about the Scale Driver to obtain the results.



*Figure 1. Input data*

**

*Figure 2. Obtained results*

The estimation results are not very diﬀerent from the previous with all “Nominal” Cost Drivers. Probably some parameters compensate other ones and the estimation result is very similar.

# Identifying the tasks and their schedule

1. RASD (Requirement Analysis Specification Document)

* Start Date 15/10/2015
* End Date  6/11/2015

1. DD (Design Document)

* Start Date 12/11/2015
* End Date 4/12/2015

1. Code Inspection Document

* Start Date 9/12/2015
* End Date 05/01/2016

1. ITPD (Integration Test Planning Document)

* Start Date 07/01/2016
* End Date 21/01/2016

1. Project Planning Document

* Start Date 22/01/2016
* End Date 02/02/2016

# 5. Allocation of members to tasks

1. RASD (Requirement Analysis Specification Document)

* Working hours per member: 40 hours each
* Tasks:
* Marija Mavcheva: Introduction, Alloy Modeling, Scenarios, Class Diagram, State Chart Diagrams
* Andrijana Mirchevska: Overall Description, Requirements, Use Case Diagrams, Sequence Diagrams.

1. DD (Design Document)

* Working hours per member: 30 hours each
* Tasks:
* Marija Mavcheva: Requirement traceability, Architectural Design, Component Interface, Algorithm Design
* Andrijana Mirchevska: Introduction, Architectural Design, Selected architectural styles and patterns, Algorithm Design

1. Code Inspection Document

* Working hours per member: 56 hours each
* Tasks:
* Marija Mavcheva: Functional role of assigned class, Method 1: createNamingContext()
* Andrijana Mirchevska: Assigned class, Method 2: addEnvironment

1. ITPD (Integration Test Planning Document)

* Working hours per member: 27 hours each
* Tasks:
* Marija Mavcheva: Integration Strategy, Tools and Test Equipment Required.
* Andrijana Mirchevska: Individual Steps and Test Description

1. Project Planning Document

* Working hours per member: 10 hours each
* Tasks:
* Marija Mavcheva: COCOMO Approach
* Andrijana Mirchevska: Functional Points

The total hours of work spent during all phases of the project are 326 hours.

326 hours/ (35∗4) hours = 2.33 Person/Months;

We suppose that a man can work 35 hours in a week so 35\*4 is a number of hours that man works in a month.

# 5. Risks and recovery actions