****

2015-2016

**Project Plan**

Morenzetti Sergio 791558.

Marcucci Antonio Pio 854473.

Released date: 02/02/2016

***Table of content:***

1. *Introduction*
2. *Function Points and COCOMO estimation*
3. *Tasks schedule*
4. *Resources allocation*
5. *Project risks*

**1 Introduction**

This document represent the Project Planning for myTaxiService Application.

Project planning is an iterative process which starts when you create an initial plan during the project startup phase.

However plan changes are inevitable: as more information about the system and the project team becomes available during the project, the plan should be regularly revised to reflect requirements, schedule and risk changes.

Changing among the goals also leads to changes in project plans, and then the project may have to be re-planned.

**2 Function Points and COCOMO estimation**

The Project Planning requires an analysis of the software effort and cost estimation, made with an Algorithmic cost modeling.

There are two different approaches: the first one is the Function Points (FP), which is used to estimate the project code size; the second is the COCOMO II that is used to estimate the efforts and costs required for the development of a project.

2.1 Function Point estimation

For this approach there is a fundamental assumption: the dimension of software can be characterized by abstraction.

This estimation is based on a combination of program characteristics:

* external inputs and outputs;
* user interactions;
* external interfaces;
* files used by the system.

For each FP type the number of its instances should be counted.

There is also a weight which is associated to each of these FP counts; the total is computed by multiplying each count by this weight, summing all partial values.

So the Function Point type are the following:

* Internal Logical File (ILF): it represents an homogeneous set of data used and managed by the application
* External Interface File (EIF): it represents an homogeneous set of data used by the application but generated and maintained by other applications.
* External Input: it represents an elementary operation to elaborate data coming from the external environment
* External Output: it represents an elementary operation that generates data for the external environment (It usually includes the elaboration of data from logic files)
* External Inquiry: it represents Elementary operation that involves input and output (Without significant elaboration of data from logic files)

Those function points, as already said, will be correlated to a corresponding weight, which changes with their complexity:

In this way we obtain the Unadjusted Function Points UFP:

Now we apply these concepts to our application, performing the calculation step by step:

* ILFs: our application stores the information about the users of myTaxiService, such as the account data (of the passengers and drivers), and also reservations data.  
  Each of these entities has a simple structure, as they are composed of a small number of fields. Thus, we can decide to adopt for all the two the simple weight.

So 2 \* 7 = 14 FPs concerning ILFs

* EIFs: our application manages the interaction with GoogleMap service, which provides us all the geographical information that myTaxiService app uses for the city zone division etc; this entity is complex because involve a lot of information, so we can adopt a complex weight for it. 1\*10=10FPs;

furthermore the app interacts with the database of the official taxi company of the city, in order to retrieve “official” drivers. This entity is simple because characterized by simple fields. 1\*5 = 5 FPs;  
  
So we have 5 + 10 = 15 FPs concerning EIFs

* External Input: the application interacts with users as follow:
* Registration: this is a simple operation, so we can adopt the simple weight for it. 1\*3=3 FPs
* Login/Logout: these are simple operation as well, so again simple weight. 2\*3=6 FPs
* Edit/Delete Account: these are simple operation and we can adopt simple weight for it. 2\*3=6 FPs
* Request/Reservation : these operations are of medium complexity because involve also geographical information, so we adopt medium weight. 2\*4=8 FPs
* Edit / delete Reservation: the first operation is of medium complexity because, as for the reservation and the request, it involves geographical information; the second is simple operation; so we can adopt medium weight and simple weight, respectively. 1\*4 + 1\*3 = 7 FPs
* Busy/ available (for a taxi driver): the first is a simple operation instead the second is a medium operation because it involves again geographical information; so we can adopt simple and medium weight respectively. 1\*3 + 1\*4 = 7 FPs
* Accept/decline a request : both operations are simple so we adopt simple weight.

2\*3 = 6 FPs

In summary we have 3+6+6+8+7+7+6= 43 FPs concerning External Input

* External Output: the application can provide a taxi in response to a passenger request; moreover it can provide a passenger to an available taxi driver. Both these operations are complex because they involve a lot of information and a difficult process; thus, we can adopt a complex weight.

So 2\*7=14 FPs concerning External Output

* External Inquiry: the application allows users to visualize their account page and, in particular for the passengers, to see the summary of their reservations. Both are simple operations, thus we adopt simple weight.

So 2 \* 3 = 6 FPs concerning External Inquiry

In summary, we have computed the following value for the unadjusted FPs: 102.

This value can be used directly 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 (Kilo lines of code) and then use another approach, such as COCOMO, to estimate the effort.

2.2 COCOMO estimation

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

characteristics of the product but also of people and process.

To pass from FP to SLOC (stands for Source Lines Of Code) we use this formula:

SLOC = AVC \* number of function points

where AVC is a language-dependent factor; in our case the average conversion factor value is 46, due to the fact that we use J2EE.

A first estimation is based on FPs approach converted to SLOC:

SLOC = 46 \* 102 = 4692

COCOMO model makes its estimates of required effort (measured in Person-Months – PM) based primarily on our estimation of the software project's size (as measured in thousands of SLOC – KSLOC):

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

where:

* EAF: Effort Adjustment Factor derived from Cost Drivers (product of the effort multipliers corresponding to each of the cost drivers for your project).
* E: Exponent derived from Scale Drivers.

Since we consider our project characterized by all “Nominal” Cost Drivers and Scale Drivers, we will have an EAF of 1.00 and exponent E of 1.0997.

So:

Effort = 2.94 \* 1.00 \* (4.692)1.0997  = 16.09 PM

Now we are going to predict the number of months required to complete our software project, using the schedule equation, where its duration is based on the efforts calculated previously:

Duration = 3.67 \* (Effort)SE

where SE is the schedule equation exponent derived from the five Scale Drivers, which in our case is 0.3179, because we considered the “Nominal” Scale Drivers.

So:

Duration = 3.67 \* (16.09)0.3179 = 8.876 Months

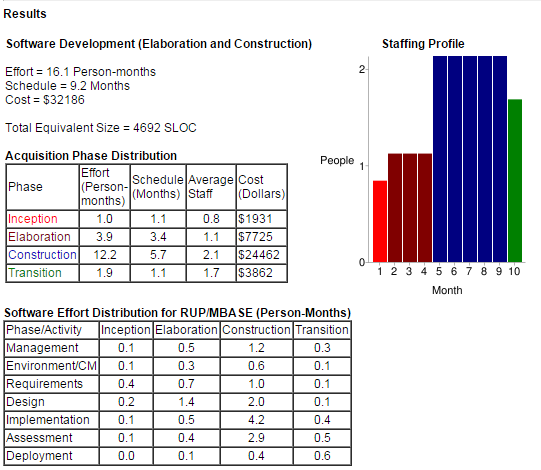
Now we can estimate the number of people needed to complete the project with the following formula:

Npeople = Effort/Duration = 16.09/8.876 = 1.81 🡪 2 people

We noticed that the estimation obtained applying COCOMO approach is quite oversized, but it is not so far from the reality. This fact could be linked to parameters value of COCOMO or maybe to the Funcion Point estimation, which could be not so realistic.

To better evaluate the effort and cost of the software project we also used an online tool that helps us to do some calculus (<http://csse.usc.edu/tools/COCOMOII.php>).

We added below the report of that site:

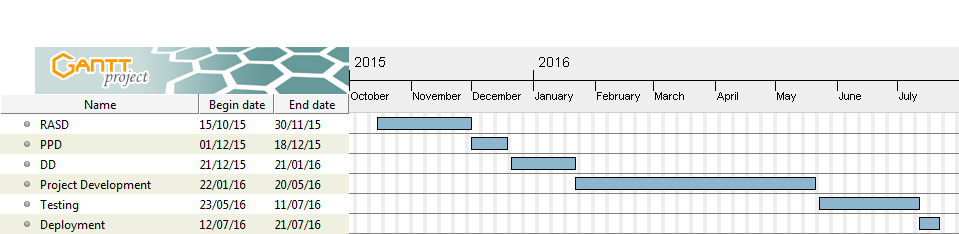


This estimation result is not very different from the previous one, obtained without the usage of any tool.

**3 Tasks Schedule**

In this section we identify the different tasks of myTaxiService project and their schedule.

|  |  |  |
| --- | --- | --- |
| **TASK** | **DURATION** (days) | **DESCRIPTION** |
| RASD | 46 | All the activities which concern the Requirements phase, such as identification of the stakeholders, elaboration of functional and non functional requirements etc. |
| PPD | 20 | Identification of the Function Point, effort estimation using COCOMO II methods and the other activities exploited in the Project Planning Document |
| DD | 30 | Identification and then analysis of the main architectural and design principles of the system, and the other related activity for the Design Document realization |
| Project Development | 120 | Development of the software |
| Testing | 49 | Unit and Integration Testing of the software |
| Deployment | 9 | First official release of the Software |

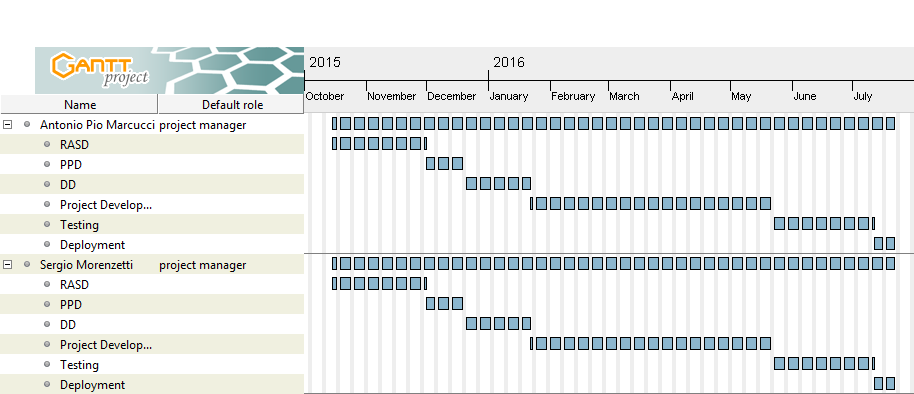
These tasks identified above should be distributed in the following way: 

We assume that each task is in-dependency with the previous one, temporally speaking.

The duration of each different task is in accordance with the temporal estimation of COCOMO.

NOTE: the date are expressed in dd/mm/yy.

**4 Resources allocation**

This graphic shows how project resources are allocated : 

The workload is equally distributed among the two members, even thought is not an optimal choice in terms of efficiency.

**5 Project risks**

In this section we define the risks for myTaxiService project, their relevance and the associated recovery actions.

In the table below we list the main risk categories, adopting three different values of relevance:

LOW: it means that the risk taken into account is not so relevant

MODERATE: the risk has medium relevance

HIGH: the risk has to be seriously taken in consideration

|  |  |  |
| --- | --- | --- |
| **RISK** | **RELEVANCE** | **RECOVERY ACTIONS** |
| Organizational financial  problems | MODERATE | Prepare a briefing document for senior management  showing how the project is making a very important  contribution to the goals of the business, presenting also  reasons why cuts to the project budget would not be cost-effective |
| Staff Recruitment problem | HIGH | Alert customer about potential difficulties and possibility of delays; investigate buying-in components |
| Staff Illness | MODERATE | Organize the team so that there is more overlap of work, in this way people should better understand each other’s jobs |
| Requirements changes | MODERATE | Derive traceability information to assess requirements  change impact; furthermore it’s important to maximize information hiding in the design, in order to protect the code from the changes |
| Organizational Restructuring | LOW | Prepare a briefing document for senior management describing how the project is working, and how it is going to accomplish the goals within deadlines |
| Database Performances | MODERATE | Consider the possibility of buying a more efficient database for the system |
| Underestimated  development time | MODERATE | Consider the possibility of adding a “middle” release of the software, and delay the final deadline; Otherwise consider the possibility of adopt buying-in components |