myTaxiService

Project **P**lan **D**ocument

A.Y. 2015/2016

Politecnico di Milano

Version 1.0

Cattaneo Michela Gaia, matr. 791685
Barlocco Mattia, matr. 792735

February 2, 2016

Contents

1 Introduction

1.1 Purpose

This document represents the Project Plan Document.
It specifies the effort and cost estimation of the myTaxiService project, by using the Function Points and COCOMO algorithmic techniques. Moreover, there is the explanation of the organization of the resources and of the tasks needed.
The intended audience of this document is the project stakeholders and the development team.

1.2 Definitions and abbreviations

- **Definitions**

- **User**: a person who requests a service from the system. It can be a visitor or a passenger.

- **Visitor**: a person who is not registered in the application.

- **Passenger**: a person who is registered in the application.

- **Taxi driver**: a taxi driver who access the application with a specific ID.

- **Request**: the request of a taxi in a certain area and position in the city made by a user.

- **Reservation**: the reservation of a taxi in a certain area, place and time that can be made only by passengers.

- **Acronyms and abbreviations**

- **RASD**: Requirement Analysis and Specification Document

- **DD**: Design Document

- **JVM**: Java Virtual Machine

- **FP**: Function Points

- **COCOMO**: COnstructive COst MOdel

- **ILF**: Internal Logic Files

- **EIF**: External Interface Files

- **KSLOC**: Kilo Source Lines Of Code

- **EAF**: Effort Adjustment Factor

1.3 Reference documents

- Project Description and Rules (https://github.com/MichelaCattaneo/myTaxiService/blob/master/Project%20Description%20And%20Rules.pdf)
- Requirements Analysis and Specification Document (https://github.com/MichelaCattaneo/myTaxiService/blob/master/Deliveries/RASD_1.1.pdf)
- Design Document (https://github.com/MichelaCattaneo/myTaxiService/blob/master/Deliveries/DD.pdf)
- Code Inspection Document(https://github.com/MichelaCattaneo/myTaxiService/blob/master/Deliveries/CodeInspection.pdf)
- Integration Test Document (https://github.com/MichelaCattaneo/myTaxiService/blob/master/Deliveries/ITPD.pdf)

2 Effort and Cost Estimation

2.1 Function Points

Function points is part of the algorithmic techniques for cost and estimation modeling. It bases on the assumption that the dimension of software can be characterized by abstraction.
This algorithmic technique is based on the complexity of five different entities:

- number of input types: concern elementary operations that elaborate the data coming from the external environment.
- number of output types: concern elementary operations that generate the data for the external environment.
- number of inquiry types: concern input that controls the execution of the program and does not change the internal data structure, according to the query criteria.
- number of internal logic files (ILF): concern the internal data generated by the system.
- number of external interface files (EIF): concern the external interfaces to other systems or applications.

Their weights are showed in the table below.

Function Types	**Weight**		
	Simple	Medium	Complex
N. Inputs	3	4	6
N. Outputs	4	5	7
N. Inquiry	3	4	6
N. ILF	7	10	15
N. EIF	5	7	10

2.1.1 External Inputs

The system allows the customer or the taxi driver to:

- Sign up
- Log in and log out
- Request a taxi
- Reserve a taxi
- Choose the payment method
- Manage the profile
- Change status
- Answer to taxi requests

All of these operations are rather simple and involves at most two entities, so a simple weight can be adopted.

Inputs: 9 x 3 = 27 FPs

2.1.2 External Outputs

The system sends to the customer information about the taxi driver arrival, after the request and to the taxi driver his position in the queue and the request notification.
The first two entities are complex, as they should get some information both from the ILF and the EIF, so a complex weight will be adopted. While for the request notification it is necessary to get information from two tables of the ILF, so it can be considered a medium complexity.

Outputs: 2 x 7 + 1 x 5 = 19 FPs

2.1.3 External Inquiry

The system allows the users and the taxi drivers to see their profile, which can be consider a simple information request.
The passenger in particular can also see his previous requests and the taxi driver can see his position in the queue of his area and the previous notifications. The visualization of the position can be considered simple, while the visualization of the requests and notification requires to load a more consistent amount of data, so they will be considered medium weighted.

Inquiries: 2 x 3 + 2 x 4 = 14 FPs

2.1.4 ILF

The application stores information about:

- passengers
- taxi drivers
- city areas
- taxi queues
- requests
- reservations

All these entities has a simple structure in a database, in fact their table is composed of a few fields. Therefore we can decide to use a simple weight for all of them.

ILF: 6 x 7 = 42 FPs

2.1.5 EIF

The application manages the interaction with three external systems:

- Google Maps, in order to determine the position of the taxi drivers and of the users.
- Facebook, in order to get the user personal information if he wants to log in with his Facebook account.
- Google+, in order to get the user personal information if he wants to log in with his Google+ account.

The position of all the different users can be considered a medium complexity entity, so we will adopt a medium weight. As regards the other two entities about the personal information, they can be considered simple weighted, as they have a simple structure.

EIF: 1 x 7 + 2 x 5 = 17 FPs

2.1.6 Result

The total number of function points is:

Function Types	**FPs**
N. Inputs	27
N. Outputs	19
N. Inquiry	14
N. ILF	42
N. EIF	17
Total FPs	119

2.2 COCOMO

COCOMO is a cost estimation model. It is based on the function points previously calculated and on the lines of code of the project, that, along with the scale drivers, help to determine the general effort, duration and number of people needed for the project.
In order to calculate the Source Lines Of Code (SLOC), it necessary to know how much SLOC per FP are needed in average for a J2EE project. This information can be found on the QSM website (`http://www.qsm.com/resources/function-point-languages-table`), where we can find the factor 46. Therefore the estimated number of lines of code is:

$$SLOC = FP * 46 = 119 * 46 = 5474$$

We can consider the Nominal values of Cost Drivers (EAF = 1.00) and Scale Drivers (E = 1.0997) in order to calculate the effort, inserting this values in the formula:

$$effort = 2.94 * EAF * (KSLOC)^E$$

$$effort = 2.94 * 1.00 * (5.474)^{1.0997} = 19.066 Person/months$$

The duration of the project can be found with this formula, considering the exponent E = 0.3179:

$$Duration = 3.67 * (effort)^E$$

$$Duration = 3.67 * (19.066)^{0.3179} = 9.37 Months$$

Now it is possible to calculate the number of people needed for this project.

$$N.people = effort/Duration$$

$$N.people = 19.066/9.37 = 2,034 people$$

2.3 Conclusions

In the Function Points part, we have overestimated some weight of the entities in order to reach an estimation that allows to finish in time or before the expected time. As regards the COCOMO results, the number of people coincide with the reality and the duration of the project is reasonable, tho maybe it is overestimated, as said before, and it is possible to reduce it.

3 Task Scheduling

We divide our project in the most relevant tasks and we give them an estimate of the time that we should spend:

- **T1** RASD: this is the first task to do. It describes what exactly we have to do: requirements, interfaces, model of the Database and how our application should work. Without this document we can't start the project.
  We estimate about 10 days to complete this task.
  Dependencies: -.

- **T2** Design Document: it describes the architectural design of our system. This task has to be done after the RASD and before the project starts.
  We estimate about 10 days to complete this task.
  Dependencies: T1.

- **T3** Database: based on the model descried in the RASD and in the Design Document, in this task we create all the tables in the database.
  We estimate about 15 days to complete this task.
  Dependencies: T2.

- **T4** Interfaces: based on what we wrote in the RASD and in the Design Document, in this task we write the code of the interfaces for the client side.
  We estimate about 2 months to complete this task.
  Dependencies: T2.

- **T5** Requirements: in this task we develop the logic of the application on the server side (for example: the AccessManager component that manages the login and the registration, the RequestManager that manages the requests, the reservations and the notifications).
  We estimate about 2 months to complete this task.
  Dependencies: T3.

- **T6** Code Inspection: when the application is done, we can check that the code is well structured.
  We estimate about 1 month to complete this task.
  Dependencies: T4,T5.

- **T7** Code Inspection Document: when the code inspection is done, we write the document of it.
  We estimate about 5 days to complete this task.
  Dependencies: T6.

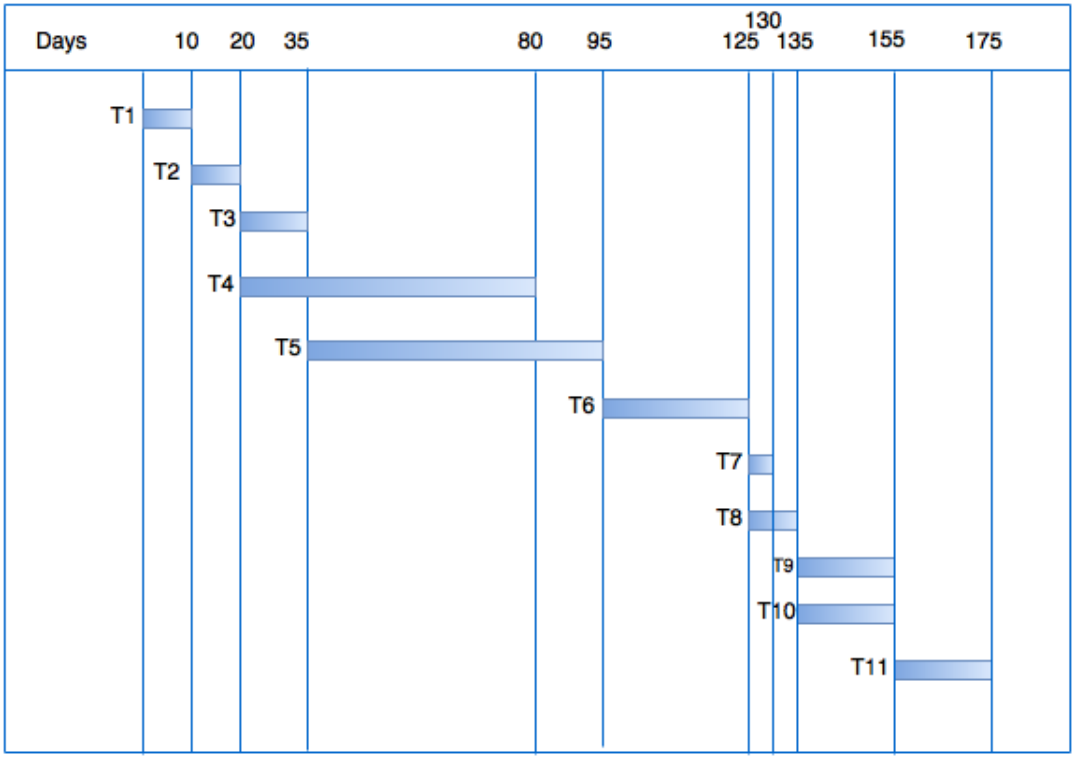
- **T8** Test Document: before starting the testing of the entire code, in this task we have to write the document related on the strategy we will adopt for the unit testing, the integration testing and the system testing.
  We estimate about 10 days to complete this task.
  Dependencies: T6.

- **T9** Unit Testing: based on what we wrote on the test document, in this task we do the unit testing.
  We estimate about 20 days to complete this task.
  Dependencies: T8.

- **T10** Integration Testing: based on what we wrote on the test document, in this task we do the integration test.
  We estimate about 20 days to complete this task.
  Dependencies: T8.

- **T11** System Testing: based on what we wrote on the test document, in this task we do the system testing.
  We estimate about 20 days to complete this task.
  Dependencies: T9,T10.

The project starts on October 2015 and based on what we have estimated and how we allocate the resources to all the tasks, we should finish the project in about 6 months on March 2016. COCOMO estimates that we should finish the project in about 9 months on June 2016, but this tool can overestimate the cost of this project because it is based on mathematical models.
This is an example of our task schedule:



4 Resources Allocation

Task	Effort(person)	Dependencies
T1	2	-
T2	2	T1
T3	1	T2
T4	1	T2
T5	1	T3
T6	2	T4,T5
T7	1	T6
T8	1	T6
T9	1	T8
T10	1	T8
T11	2	T10,T9

Based on our schedule and the availability of our resources, the members of the team can be allocated to the tasks in this way:

- **T1**: we can allocate all the resources (two people) to this task because without it we cannot start the next tasks. In fact we allocate all the resources to the RASD to accelerate it and to start as soon as possible the other tasks.
- **T2**: we allocate all the resources (two people) to the Design Document because it continues the description of our project.
- **T3**: we allocate one person to the creation of the database after the T2.
- **T4**: we allocate one person to the development of the interfaces during the creation of the database because this task don't need the database.
- **T5**: we allocate one person to the development of the server side of the system (requirements). The person that we allocate to this task is the same that we allocated to the task 3 because this task needs a database to be developed.
- **T6**: when T4 and T5 are done, we allocate all the resources (two people) to this task because it is a long work and the time can be reduced with two people.
- **T7**: after T6, we allocate one person to write the code inspection document.
- **T8**: after T6 and during T7, we can allocate the other person to write the test document.
- **T9**: after T8, we allocate one person to do the unit test.
- **T10**: during T9, the other person can be allocated to the integration test.
- **T11**: when T9 and T10 are done, we allocate all the resources for the last test.

5 Risks and Recoveries

Risk management is an important issue for the development of a project.
The first step consists in identifying potential problems and threats the product is subject to. Then it is necessary to determine the recovery actions and all the ways to reduce them.
These are the risks that could be found:

- **Project risk.** It could be difficult to follow the project schedule and therefore the product is finished later than the estimated time. This will imply an increase in the efforts and in the costs.
  In order to avoid this kind of problem, which has a high relevance, it is necessary to monitor every phase of the project, comparing it with existing projects.

- **Business risk**. In particular market risk, in fact it is possible that, even tho the project is good and efficient, it is difficult to sell because it is not what people needs, for example if there is already a similar product preexisting.
  There is not a real strategy to adopt against this kind of risk, in fact it is quite unpredictable and depends on market trends, in order to try to reduce the possibility of this kind of situations a study on the market could help.

- **Technology to be built.** This is a critical risk: it is possible that, going on in the implementation of the system, the requirements grows with respect to the one identified in the beginning.
  A possible strategy is to overestimate the project in the early phases and anticipate all the possible scenarios.

- **Staff size and experience.** It is possible that the lack of technical experience of the project developers affect the quality of the software and the schedule of the project.
  In order to prevent this from happening it is necessary that at least one component of the group has experience on previous projects or that the group follows existing guidelines. This is not a really relevant risk, as the project developers can always take into account previous projects and material.

It is important to be ready to recover from any potential risk, as long as they can affect the project schedule or the quality of the product, increasing the costs and efforts that were previously estimated.
The strategy adopted is the proactive strategy, that follows fixed steps in order to manage the risk in advance, instead of worrying about the problems only afterwards. In fact, it is important to take a forward-looking view, thinking about all the future issues that may arise.
Any possible threat is identified and analyzed, then there is an estimation of the probability and the impact of the damage. This analysis creates a ranking of all the risks, that allows to create a contingency plan that handles the higher priority and impact risks first.

6 Appendix

6.1 Software and tool used

- TeXstudio (`http://www.texstudio.org/`): to redact and to format this document.

6.2 Hours of work

Time spent redacting this document:

- Cattaneo Michela Gaia: ~_ hours of work.
- Barlocco Mattia: ~_ hours of work.