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 Function Points and COCOMO algorithmic techniques. Moreover, it is given an explanation of the organization of the resources and of the tasks needed, along with the list of all the risks it is possible to encounter.
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

- **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 the 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.

The weights of this entities 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 involve 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. It sends to the taxi driver his position in the queue and the requests 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:

- passenger
- taxi driver
- city area
- request
- reservation

All these entities, except for the city area, have a simple structure in the database, in fact their table is composed of a few fields. Therefore we can decide to use a simple weight for all of them. As regards the city area table, it can be considered of medium complexity because it has to store also the information about the taxi queues.

ILF: 4 x 7 + 1 x 10= 38 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, Facebook and Google+, 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	38
N. EIF	17
Total FPs	115

2.2 COCOMO

COCOMO is a cost estimation model. It is based on the FP 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:

$$SLOC = FP * 46 = 115 * 46 = 5290$$

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:

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

$$effort = 2.94 * 1.00 * (5.29)^{1.0997} = 18.36 Person/months$$

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

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

$$Duration = 3.67 * (18.36)^{0.3179} = 9.25 Months$$

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

$$N.people = effort/Duration$$

$$N.people = 18.36/9.25 = 1.98 people$$

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, considering the number of hours we have calculated for the documentation and the time needed for the code and testing.

3 Task Scheduling

Here is presented a list of the relevant tasks of the project, with their schedule, dependencies and the time spent for each one:

- **T1** RASD: this is the first task to accomplish. It describes the most important features of the project: requirements, assumptions, interfaces, model of the Database and how the myTaxiService application should work. It is necessary to redact and finish this document, identifying all the essential aspects, before it is possible to start the project.
  Estimated time: about 10 days.
  Dependencies: -.

- **T2** Design Document: this document describes the architectural design of the myTaxiService system. It is also part of the documentation necessary to finish before the project starts. In facts, after the RASD has been done, it is necessary to identify the architecture and the components of the project, before actually starting to code.
  Estimated time: about 10 days.
  Dependencies: T1.

- **T3** Database: based on the model described in the RASD and in the Design Document, this task consists in the creation of the tables of the database.
  Estimated time: about 15 days.
  Dependencies: T2.

- **T4** Interfaces: based on the interfaces shown in the RASD and in the Design Document, this task contains the code implementing the interfaces on the client side.
  Estimated time: about 2 months.
  Dependencies: T2.

- **T5** Requirements: this task consists in the development of the application logic on the server side (for example the AccessManager component that manages the login and the registration or the RequestManager that manages requests, reservations and notifications).
  Estimated time: about 2 months.
  Dependencies: T3.

- **T6** Code Inspection: once the code part is completed, it is necessary to check that the code is bug free, also verifying that all the coding conventions are respected, in order to improve its quality and, at the mean time, to assure that there are no issues.
  Estimated time: about 1 month.
  Dependencies: T4,T5.

- **T7** Code Inspection Document: once the code inspection is done, it is necessary to document any issue found in the code.
  Estimated time: 5 days.
  Dependencies: T6.

- **T8** Test Document: this task consists of the redaction of a document that explains the strategy adopted for unit testing, integration testing and system testing. This must be done, after the code is completed and before the actual testing.
  Estimated time: about 10 days.
  Dependencies: T6.

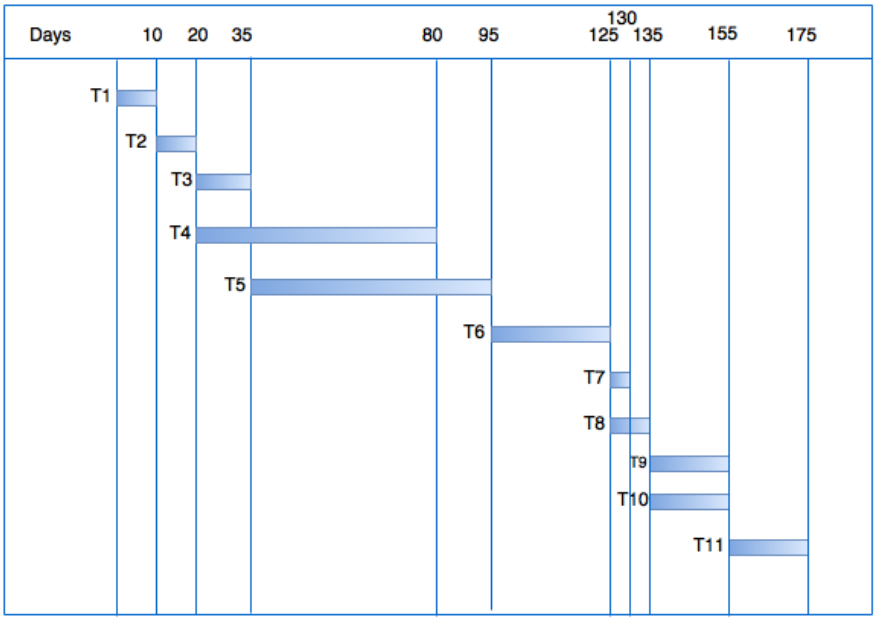
- **T9** Unit Testing: according to what is written in the Test Document, this task involves accomplishing the unit testing.
  Estimated time: about 20 days.
  Dependencies: T8.

- **T10** Integration Testing: according to what is written in the Test Document, this task involves accomplishing the integration testing.
  Estimated time: about 20 days.
  Dependencies: T8.

- **T11** System Testing: according to what is written in the Test Document, this task involves accomplishing the system testing.
  Estimated time: about 20 days.
  Dependencies: T9,T10.

The project should start on October 2015 and, according to the time estimation and the resource allocation, it should be over in about 6 months on March 2016. Comparing this assumption with the COCOMO results, that states that the project should last 9 months, the reality could reasonably be in the middle of this two estimation.

This is a simple Gantt chart of the 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 the 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 the resources are allocated to the redaction of the RASD in order to accelerate its completion and therefore start as soon as possible the other tasks.
- **T2**: we allocate all the resources (two people) to the redaction of the Design Document because it is necessary to complete the architectural design before continuing with the project development.
- **T3**: we can allocate one person to the creation of the database after the task T2.
- **T4**: we can allocate one person to the development of the interfaces during the creation of the database, because this task does not need the database implementation part.
- **T5**: we can 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**: once T4 and T5 are done, we allocate the two people in the team to this task because it requires a good deal of time and its duration can be reduced with two people.

- **T7**: after T6, we allocate one person to redact 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 write unit tests.
- **T10**: during T9, the other person can be allocated to the integration test.
- **T11**: once T9 and T10 are done, we allocate all the resources for the system test.

5 Risks and Recoveries

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

- **Project risk.** This is a critical risk: it is possible that, going on in the implementation of the system, the requirements unexpectedly grows with respect to the one identified at the beginning. So 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. In case this is not enough, it can always be possible to release an early version of the project with less functionalities and then, when possible, release the complete version.

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

- **Technology to be built.** This is a relevant risk relied to the technological part of the application: the most critical issues for an application like myTaxiService, that should be reliable and responsive, are data loss and performance issues.
  In order to avoid the deterioration of the service, a particular care on this aspect must be taken in account. The server should always be on and available, in order to compute all the requests in a short amount of time and the algorithms computing the taxi queues and the taxi positioning should be implemented in the most efficient ways. In particular, to prevent data loss, the functioning protocol for the client-server data transmission should be adopted.

- **Staff 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 the group is guided by someone with more experience or that the developers are able to follows existing guidelines of previous projects. This is not a really relevant risk, as the project developers can always take into account previous projects and material.

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: ~12 hours of work.
- Barlocco Mattia: ~12 hours of work.