myTaxiService

-

Project Plan Document

Davide Cremona (matr. 852365), Simone Deola (matr. 788181)

January 27, 2016

Contents

Chapter 1

Introduction

1.1 Purpose and Scope

1.1.1 Purpose

The purpose of this document is to provide an estimation of how big is the project and how much time will take to develop and release the entire system. Also, we will provide a list of tasks that need to be executed and the schedule of execution for the project.

1.1.2 Scope

The scope of the developed software is to provide an interface between the customers and the taxi drivers. The application is in charge to simplify the requests and the reservations of the rides and the management of the city queues.

1.2 List of Definitions and Abbreviations

1.2.1 Definitions

- **Taxi Drivers**: Taxi Drivers are the workers that drive a taxi. They are also users of myTaxiService.
- **Customers**: Customers are the end-users of the application.
- **Client**: Client side of the system.
- **Server**: Server side of the system.
- **Team Member**: project managers, designers and developers of the application.

1.2.2 Abbreviations

- **DD**: State for Design Document
- **RASD**: States for Requirement Analysis and Specification Document.
- **ITPD**: Integration Test Plan Document.
- **GPS**: Global Positioning System.
- **FP**: Functional Points.
- EAF: Effort Adjustment Factor.
- $SLOC$: Source Line of Code.
- $KSLOC$: Kilo Source Line of Code (thousands of SLOC).
- **COCOMO**: COnstructive COst MOdel.

1.3 List of Reference Documents

- **Application Description**: downloadable from here: `https://github.com/SimoneDeola/CremonaDeola/blob/master/source/testPlan/documents/Assignments%201%20and%202%20(RASD%20and%20DD).pdf`
- **RASD**: downloadable from here: `https://github.com/SimoneDeola/CremonaDeola/raw/master/Deliveries/%5B1%5Drasd.pdf`
- **DD**: downloadable from here: `https://github.com/SimoneDeola/CremonaDeola/raw/master/Deliveries/%5B2%5Ddd.pdf`

Chapter 2

Project Size, Effort and Cost

2.1 Project Size

The purpose of this section is to estimate Function Points to give an estimation of the project size. We will use this Size Estimation Procedure:

- Determine the function counts by type: Count the number of functions for each Function Type.
- Determine the complexity level for each Function Type.
- Apply weights to the Function Types.
- Compute the Function Points for each Function Type.

Each subsection will take in account a different User Function Type. User Function Types are described in the following table:

External Input (Inputs)	Count each unique user data or user control input type that (i) enters the external boundary of the software system being measured and (ii) adds or changes data in a logical internal file.
External Output (Outputs)	Count each unique user data or control output type that leaves the external boundary of the software system being measured.
Internal Logical File (Files)	Count each major logical group of user data or control information in the software system as a logical internal file type. Include each logical file (e.g., each logical group of data) that is generated, used, or maintained by the software system.
External Interface Files (Interfaces)	Files passed or shared between software systems should be counted as external interface file types within each system.
External Inquiry (Queries)	Count each unique input-output combination, where an input causes and generates an immediate output, as an external inquiry type.

Table 2.1: Function Types

To determine the complexity level of each Function Type, it's used the following tables:

For ILF and EIF			
Record	**Data Elements**		
Elements	**1 - 19**	**20 - 50**	**51+**
1	Low	Low	Average
2 - 5	Low	Average	High
6+	Average	High	High

Table 2.2: External Inputs and External Interface Files complexity distribution

For EO and EQ			
Record	**Data Elements**		
Elements	**1 - 5**	**6 - 19**	**20+**
0 or 1	Low	Low	Average
2 - 3	Low	Average	High
4+	Average	High	High

Table 2.3: External Output and External Inquiries complexity distribution

For EI			
Record Elements	**Data Elements**		
	1 - 4	**5 - 15**	**16+**
0 or 1	Low	Low	Average
2 - 3	Low	Average	High
3+	Average	High	High

Table 2.4: External Inputs complexity distribution

To determine the weights for each Function type, the following table has been used (for each Function Type, a weight is assigned):

Weights			
Function Type	**Complexity-Weight**		
	Low	**Average**	**High**
Internal Logical Files	7	10	15
External Interface Files	5	7	10
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6

Table 2.5: Function Types Weights

2.1.1 Internal Logical Files

Internal Logical Files are stored for:

- **Users (Customers and Taxi Drivers)**
  Users have to memorize from 8 to 5 elements and for each user there are two external records that have to be memorized (Position and Zone). So, according to the table 2.2, the complexity of "Users" is LOW and, according to the table 2.5 the Functional Points assigned are 7.

- **Rides (Reservations and Single Rides)**
  Rides have to memorize from 3 to 1 elements and for each ride there are three external records that have to be memorized (Position, Customer and Taxi Driver). So, according to the table 2.2, the complexity of "Rides" is LOW and, according to the table 2.5 the Functional Points assigned are 7.

- **City Zones**
  City Zones have to memorize from 3 to 5 elements and for each user there are one external record that have to be memorized. So, according to the table 2.2, the complexity of "City Zones" is LOW and, according to the table 2.5 the Functional Points assigned are 7.

The computation of the final Functional Points assigned to the class "Internal Logical Files" is:
FP(ILF) = 7+7+7 = 21.

2.1.2 External Interface Files

External Interface Files are stored for:

- **GPS**
  Informations about the GPS position of the users are essentialy represented as a "Position" (Latitude and Longitude). So according to the table 2.2 the complexity is LOW and according to the table 2.5 the Functional Points assigned are 5.

- **Facebook Accounts**
  Informations about the users that comes from the Facebook Login System, are represented as Customers (5 attributes). The external records are two (Position and Zone). So According to the table 2.2 the complexity is LOW and according to the table 2.5 the Functional Points assigned are 5.

The computation of the final Functional Points assigned to the class "External Interface Files" is:
FP(EIF) = 5+5 = 10.

2.1.3 External Inputs

External Inputs are:

- **Registration (of a Customer, with Facebook and of a Taxi Driver)**
  These operations involve only one data type (User) and involve less than 5 elements. According to the table 2.4 the complexity is LOW and the Functional Points assigned to each operation is 3. The overall Functional Points assigned are: FP(Registration) = 3 x 3 = 9.

- **Login/Logout**
  These operations involve only one data type (User) and involve less than 5 elements. According to the table 2.4 the complexity is LOW and the Functional Points assigned to each operation is 3. The overall Functional Points assigned are: FP(Login/Logout) = 3 x 2 = 6.

- **Profile Modification (Availability or Details modification)**
  These operations involve only one data type (User) and involve less than 5 elements. According to the table 2.4 the complexity is LOW and the Functional Points assigned to each operation is 3. The overall Functional Points assigned are: FP(Profile Modification) = 3 x 2 = 6.

- **Report Abuse (Customer or Taxi Driver)**
  These operations involve only one data type (User) and involve less than 5 elements. According to the table 2.4 the complexity is LOW and the Functional Points assigned to each operation is 3. The overall Functional Points assigned are: FP(Report Abuse) = 3 x 2 = 6.

- **Taxi Request** These operations involve only one data type (Ride) and involve less than 15 elements. According to the table 2.4 the complexity is LOW and the Functional Points assigned to each operation is 3. The overall Functional Points assigned are: FP(Taxi Request) = 3 x 1 = 3.

- **Taxi Reservation/Reservation Deletion** These operations involve only one data type (Ride) and involve less than 15 elements. According to the table 2.4 the complexity is LOW and the Functional Points assigned to each operation is 3. The overall Functional Points assigned are: FP(Report Abuse) = 3 x 2 = 6.

- **Accept/Decline Ride** These operations involve two data types (Taxi Driver (priority) and Ride) and involve less than 15 elements. According to the table 2.4 the complexity is AVERAGE and the Functional Points assigned to each operation is 3. The overall Functional Points assigned are: FP(Accept/Decline Ride) = 7 x 1 = 7.

The total Functional Points assigned to the class "External Inputs" are:
FP(EI) = FP(Accept/Decline Ride) + FP(Report Abuse) + FP(Taxi Request) + FP(Report Abuse) + FP(Profile Modification) + FP(Login/Logout) + FP(Registration) = 43

2.1.4 External Inquiries

Functionalities that causes External Inquiries are:

- **View Reservations** This functionality involve only one data type (Rides) and can potentially involve more than 20 elements. According to the table 2.3 the complexity is AVERAGE and according to the table 2.5 the Functional Points assigned are 4.

- **View Profile Informations** This functionality involve only one data type (User) and can involve from 5 to 8 elements. According to the table 2.3 the complexity is LOW and according to the table 2.5 the Functional Points assigned are 3.

The computation of the final Functional Points assigned to the class "External Inquiries" is:
FP(EI) = 4+3 = 7.

2.1.5 External Outputs

The operations that generate External Outputs are:

- **Ride Request Notification**
  For this operation are involved two data types (Zones, Users) and are considered from 6 to 19 elements. According to the table 2.3 the complexity is AVERAGE and according to the table 2.5 the Functional Points assigned are 5.

- **Incoming Taxi Notification**
  For this operation are involved two data types (Ride Request, Users) and are considered from 6 to 19 elements. According to the table 2.3 the complexity is AVERAGE and according to the table 2.5 the Functional Points assigned are 5.

The computation of the final Functional Points assigned to the class "External Outputs" is:
FP(EO) = 5+5 = 10.

2.1.6 Computation of Total Functional Points

The final computation of the Functional Points is:
FP = FP(ILF) + FP(EIF) + FP(EInputs) + FP(EInquiries) + FP(EO) = 21 + 10 + 43 + 7 + 10 = 91.

For Line of Code count, we consider two main programming languages: Java and C++. The computation of the Lines of Code in functions of the Functional Points is:

- **Java** (SLOC/FP = 53) is: TotalSLOC = 91 x 53 = 4823.
- **C++** (SLOC/FP = 55) is: TotalSLOC = 91 x 55 = 5005.

2.2 Project Effort and Cost

To make an Effort estimation we have to compute this formula:

$$effort = 2.94 * EAF * (SLOC/1000)^E$$

Where:

- $effort$ is the value to estimate.
- 2.94 is a constant defined by COCOMO II 2000.0.
- EAF is the "Effort Adjustement Factor" calculated in the following sections.
- $SLOC/1000$ that is the KSLOC (Kilo SLOC) number.
- E that is an exponent calculated in the following sections.

2.2.1 Computation of E parameter

The parameter E is calculated as:

$$E = B + 0.01 * SFs$$

Where:

- B is a constant given by the COCOMO II and his value is 0.91.

- SFs is an aggregation of five Scale Factors described in the document at `http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf`

The following table contains the variuos values for these Scale Factors (table taken from the previous URL).

Table 10. Scale Factor Values, SF_j, for COCOMO II Models

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	thoroughly familiar
SF_j:	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
SF_j:	5.07	4.05	3.04	2.03	1.01	0.00
RESL	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
SF_j:	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
SF_j:	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	The estimated Equivalent Process Maturity Level (EPML) or					
	SW-CMM Level 1 Lower	SW-CMM Level 1 Upper	SW-CMM Level 2	SW-CMM Level 3	SW-CMM Level 4	SW-CMM Level 5
SF_j:	7.80	6.24	4.68	3.12	1.56	0.00

- **PREC** (Previous experience of the organization with these type of projects): LOW (4.96).
- **FLEX** (Degree of flexibility in development process): HIGH (2.03).
- **RESL** (Risk Analysis): HIGH (2.83).
- **TEAM** (How well the development team knows each other): EXTRA HIGH (0.00).
- **PMAT** (Maturity of the Organization): NOMINAL (4.68).

The result is the sum of these factors:

$$SFs = \sum_{j=1}^{5} SF_j = 14.67$$

So we can now compute the E parameter:

$$\begin{aligned} E &= B + 0.01 * SFs \\ &= 0.91 + 0.01 * 14.67 \\ &= 1.558 \end{aligned}$$

2.2.2 Computation of EAF parameter

All the parameters discussed in this paragraph are described in section 3.2 of the document at this URL: `http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf`

- **Required Software Reliability (RELY)** The results of a failure in our system is easily recoverable (e.g. loss of priority in the zone queues) so the RELY factor is LOW (count as 0.92).
- **Data Base Size (DATA)** The effort needed to assemble and maintain the data required to complete test of the program is such that the DATA factor is NOMINAL (count as 1.00).
- **Product Complexity (CPLX)** The area of our project is "Data Management" and the complexity is HIGH because we have simple triggers activated by data stream contents and complex data restructuring (e.g. maintain the zone queues updated) (count as 1.17).
- **Developed for Reusability (RUSE)** This factor accounts for the additional effort needed to construct components intended for reuse on current or future projects. We think that we only have to design modules for the internal reuse in the myTaxiService system, so the descriptor is: NOMINAL (count as 1.00).
- **Documentation Match to Life-Cycle Needs (DOCU)** We think that the documentation we have to produce is necessary in function of the lifecycle of the system, so we assign to DOCU the NOMINAL level (count as 1.00).
- **Execution Time Constraint (TIME)** Our system is supposed to be fast because it's a Taxi reservation system, so the system is supposed to occupy less than 50 percent of the available execution time. The label for TIME is NOMINAL (count as 1.00).
- **Main Storage Constraint (STOR)** Due to the fact that the web application does not occupy the user's memory and that the mobile application is relatively simple, we only have to consider the memory occupation on the server side. This memory occupation change in function of the number of the users and increments if the usage of the system is high. We have to consider many things like Server Software, External API implementation, Database Space, so the label for TIME is HIGH (count as 1.05).

- **Platform Volatility (PVOL)** Due to the fact that we have to implement a Mobile Application and a Web application, and the technologies we have to use (Android, iOS, Windows Phone programming languages, JS Frameworks for the website) receives continuous updates, we have to consider the PVOL facor at least NOMINAL (count as 1.00).
- **Analyst Capability (ACAP)** Through the designing of the project, the team has collaborated, designed and analyzed every different scenarios for the application. There is a VERY HIGH rating for the ACAP factor (count as 0.71).
- **Programmer Capability (PCAP)** Another time, the development team (that is the project management team) is heavily skilled in cooperation and communication because we have developed other projects with great results. The rating for PCAP is VERY HIGH (count as 0.76).
- **Personnel Continuity (PCON)** Since we are the only two persons at work on this project, we does not count PCON as a factor that determines the EAF parameter (FULL continuity is EXTRA HIGH and it has no coefficient).
- **Applications Experience (APEX)** Since we have developed another project for 6 months we have a LOW level for the APEX factor (count as 1.10).
- **Platform Experience (PLEX)** We have experience in using Databases, building GUIs and developing distributed systems and we have worked with these structures for more than 6 months. But since our experience is not the same in every subject, we prefer to set the PLEX factor as LOW (count as 1.09).
- **Language and Tool Experience (LTEX)** Our team is skilled in development and designing OO applications and Distributed systems. But we are not so much skilled in building the Documentation (Requirements Analysis, Project Plan, Integration Test Documents, etc...) so we prefer to set the LTEX factor as NOMINAL (count as 1.00).
- **Use of Software Tools (TOOL)** All the tools we have used are for the design of the application (Lifecycle tools) and for development (Code & Edit) so we have to set TOOL factor to NOMINAL (count as 1.00).
- **Multisite Development (SITE)** The team is composed by two persons, we live at a distance of about 10 Kilometers, we have worked in the same place for the entire duration of the project enabling the direct communication between us. The SITE factor is set to EXTRA HIGH (count as 0.80).
- **Required Development Schedule (SCED)** We have a fair distribution of time thanks to the things discussed in SITE factor. So we can set SCED factor as NOMINAL (count as 1.00)

Now, given these factors we can define the set of factors:

$$F = \{RELY, DATA, CPLX, RUSE,$$
$$DOCU, TIME, STOR, PVOL,$$
$$ACAP, PCAP, PCON, APEX,$$
$$PLEX, LTEX, TOOL, SITE, SCED\}$$

and we can define a generic element of the F set as:

$$f_i \text{ such that } i = 1, 2, ..., |F|$$

So the final EAF parameter can be computed as:

$$EAF = \prod_{i=1}^{|F|} f_i = 0.585$$

2.2.3 Effort and Duration

Now thate we have successfully computed the EAF and the E parameters, we can now apply the $effort$ estimation function, considering the Java implementation:

$$\begin{aligned} effort &= 2.94 * EAF * (SLOC_{Java}/1000)^E \\ &= 2.94 * 0.585 * (4823/1000)^{1.558} \\ &= 19.96\ PM \end{aligned}$$

Since the Number of Persons ($N_{persons}$) that has to be assigned to the project is calculated as:

$$N_{persons} = effort/projectDuration$$

We can exploit this formula (since the team has a fixed $N_{persons} = 2$) and calculate how much time the team needs to develop the entire system:

$$\begin{aligned} projectDuration &= effort/N_{persons} \\ &= 19.96/2 \\ &= 9.98\ months\ (10\ months) \end{aligned}$$

Chapter 3

Project Tasks and Schedule

In this chapter is described how the work is divided and organized. For each part of this project we make a brief description, we report the deadline date and what team member is assigned to that part. Our team is composed of two members, Simone Deola and Davide Cremona. Parts of the project:

- **RASD and Project Plan**: In this part the team produce a first release of the Requirements Analysis and Specification Document an a Project Plan Document in which it is described the project and the team make an estimation of what it will be done and when it will be done. We specify that this two documents are only the first releases because they can be changed on the future steps. The two documents can be done in parallel or consequentially.
  - Estimate starting time: 10/2015.
  - Deadline:6/11/2015.
  - Workers: Simone Deola, Davide Cremona.
- **Design Document**: In this part the team produce a first release of the Design Document of the system. This document describe how the system is organized. So, in this phase the team start to determine the structure of the system and how the system works. We specify that this document are only the first release because it can be changed on the future steps.
  - Estimate starting time: 7/11/2015.
  - Deadline:4/12/2015.
  - Workers: Simone Deola, Davide Cremona.
- **Integration Test Plan Document**: In this part the team produce a first release of the Integration Test Plan Document. This document describe how to test the various part of the system. We specify that this document are only the first release because it can be changed on the future steps.
  - Estimate starting time: 5/12/2015.

  - Deadline:21/01/2016.
  - Workers: Simone Deola, Davide Cremona.

- **Coding and Unity test**: At this point of the project we have described how the system is formed so each team member can take independent parts of the system and develop it. How to split the parts of the system will be more clear when the DD will be produced. The code that is developed on this part must be tested too, so each member must produce the unity test on the part that have produced. On this part the members can work in parallel since the documentation define all the interfaces and how they works. For example, Simone Deola can work on the Database and on the front end while Davide Cremona works on the back end part (if the structure of the system will be defined in this way).
  - Estimate starting time: 22/01/2016.
  - Deadline:22/06/2016.
  - Workers: Simone Deola, Davide Cremona.

- **Integration and Test Phase**: In this phase of the project the team integrate all the part of the system, developed and tested in the previous phase, and test how it works when putted together. On this phase the team release also the integration tests. The step of the integration and the relative tests will be described on the ITPD. On this phase the two members of the team will work together.
  - Estimate starting time: 23/06/2016.
  - Deadline:31/07/2016.
  - Workers: Simone Deola, Davide Cremona.

- **Implementation**: In this phase the team makes the various Performance Test of the system. After this step the system is ready to be released.
  - Estimate starting time: 01/08/2016.
  - Deadline:30/08/2016.
  - Workers: Simone Deola, Davide Cremona.

Chapter 4

Project Risks and Recovery Actions

In this chapter we analyze the various risks that can occur and give a recovery action.

4.1 Internet Connection Failure

Since our system is a distributed one, the main risk is that the communication between Client and Server cannot be done. This occur when the Internet connection fails and the Client cannot connect to the Server application. To prevent this failure we cannot do much because the main causes to this problem are related to ISP problems. The only thing we can do is to provide a stable system with a very low offline time.

4.2 GPS Not Enabled

Since the application is based on the knowledge of the taxis positions, one of the main problems is that the GPS informations are not available. This can be fixed with position triangulation using cellphone cells. The position is not precise but it's an accettable estimation. A popup notification can be shown to the user to tell him that his position can be imprecise.

4.3 System Communication Failure

Another risk that we have to take into account is that the system fails in communicating between the various parts. This problem can be fixed with a good use of programming language constructs that checks if something is wrong at runtime.

4.4 Environmental Problems and Servers Breakdown

A risk is that the servers or the machines are attacked by natural problems like storms or fire. To prevent the complete black-out of the system we have to backup all the data into servers in a different place and to copy all the system in other machines different by the original ones.

4.5 Accident Notification

Our system cannot respond if the Taxi Driver has an accident when he's going to the client. If this problem occur, the Bad Evaluated Taxi Driver can appeal to our company to modify his bad evaluation counter.

Chapter 5

Appendix

5.1 Used Software

To create this document we have used some common softwares:

- For the latex we have used two different softwares:
  - Simone Deola: TexShop, provided with the MacTex package (link)
  - Davide Cremona: Sublime Text editor (link) with LaTeXTools (link) and the Basic Package of MacTex (link)

5.2 Hours of Work

- Cremona Davide (852365): 10 Hours
- Deola Simone (788181): 10 Hours