Chapter 1

Project Estimations

In this chapter we are going to estimate the main features of *myTaxiService* project, by using COCOMO II. Reading from the reference manual:

> *The COCOMO II model is part of a suite of Constructive Cost Models. This suite is an effort to update and extend the well-known COCOMO (Constructive Cost Model) software cost estimation model originally published in Software Engineering Economics by Barry Boehm in 1981.*

In the section 1.1 we focus on the project's size in term of lines of code, whereas in the section 1.2 other metrics, such the required time and the costs will be analysed.

1.1 Project Size (Function Points)

Reading from the reference manual:

> *The function point cost estimation approach is based on the amount of functionality in a software project and a set of individual project factors [Behrens 1983; Kunkler 1985; IFPUG 1994]. Function points are useful estimators since they are based on information that is available early in the project life-cycle. A brief summary of function points and their calculation in support of COCOMO II follows.*

The function types are five, described in the table[1].

Function Point	**Description**
External Input (EI)	Count each unique user data or user control input type that enters the external boundary of the software system being measured.
External Output (EO)	Count each unique user data or control output type that leaves the external boundary of the software system being measured.
Internal Logical File (ILF)	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 (EIF)	Files passed or shared between software systems should be counted as external interface file types within each system.
External Inquiry (EQ)	Count each unique input-output combination, where input causes and generates an immediate output, as an external inquiry type.

Finally, to perform the analysis we have to present other two tables from the same reference manual of the other one. The first one will be used to classify each function on three level of complexity (high, medium and low).

The second one shows the weights to be used into the estimation formulas[2].

[1]The table is given by the COCOMO II reference manual.

[2]The UFP acronym means Unadjusted Function Points

Table 2. FP Counting Weights

For Internal Logical Files and External Interface Files

	Data Elements		
Record Elements	**1 - 19**	**20 - 50**	**51+**
1	Low	Low	Avg.
2 - 5	Low	Avg.	High
6+	Avg.	High	High

For External Output and External Inquiry

	Data Elements		
File Types	**1 - 5**	**6 - 19**	**20+**
0 or 1	Low	Low	Avg.
2 - 3	Low	Avg.	High
4+	Avg.	High	High

For External Input

	Data Elements		
File Types	**1 - 4**	**5 - 15**	**16+**
0 or 1	Low	Low	Avg.
2 - 3	Low	Avg.	High
3+	Avg.	High	High

Table 3. UFP Complexity Weights

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

Up to now, we have presented the Function Points technique. Now, we are going to start our analysis, split by the function type.

1.1.1 Internal Logic Files

The system has to manage Internal Logic Files to store information related to the users (both *normal* and drivers), the *historical* rides, the areas and the driver workshifts[3]
The users have from 12 to 16 fields to be stored (the second number is referred to the drivers case) and only the alerts and the zerotime or future rides have to be stored, thus the complexity is low. The areas and the workshifts can also be considered as low complexity type. In fact they have a few fields and less than six extra records.
The rides have 10 fields, including two positions, the driver and the passenger, all saved in separate entities. They can be considered as an average complexity type, since we have about seven records per ride (in fact in addition to the five presented, the positions requires additional records).

In the table the analysis is summarize:

ILF	Complexity	FP
User	Low	7
Area	Low	7
Workshift	Low	7
Ride	Average	10
Total		31

1.1.2 External Logic Files

The system acquires data from te GPS interface. A GPS position is essentially a tuple of type Position, described in our database. Hence, we have a low complexity type and 5 FP.

[3]See the logic schema at the page 10 of the Design Document to have a detailed description of each part of the database.

1.1.3 External Inputs

The possible interactions between the users and the system, defined in the RASD, are now quickly described in terms of complexity:

- Login/Logout: these operations are simple due to one entity only is involved, so the complexity is low;
- Start Waiting Time/End of a ride: these operation requires to interact with three types of files (Position, Area and Driver Waiting) with one element per type, thus the complexity is low;
- Check the Reservations: this is a group of three related operations (shows the alerts and gives the possibilities to modify or to cancel a ride) that involve one type and potentially more than 16 elements, so the complexity is average;

In the following table we have summarized the results:

EI	Complexity	FP
Login/Logout	Low	2x3
Start Waiting Time / End of a Ride	Low	2x3
Check the Reservations	Average	3x4
Total		24

1.1.4 External Inquiries

The system allows the user to interact with its thought the following operations:

- Registration: this operation is performed only by simple user (not a driver) and involves one data type, the one related to the new user. Its complexity is low;
- Profile Management: this operation allows the user to modify a little personal data, so the complexity is low;
- Workshift Management: this operation requires to interact to 2 entities (driver and work shift) and can involved more than 20 elements to perform the validity checks. Hence the complexity is high;
- Book a ride (both future or zerotime): these operations needs many inter-

actions between the system and the user, for instance to show the detected position or to analyse the inserted time/address;

- Ride Allocation: we also insert this function because the system has to interact in a simple way with the taxi driver. Since many and many taxi drivers or queues may be involved before one is available, the complexity is high. Note that the notifications of the operations are not reported here, but in the External Input section;
- Taxi Driver Ride Request: this is the operation used to ask a rider the availability for a ride, the complexity is low.

In the following table we have summarize the results:

EQ	Complexity	FP
Registration	Low	3
Profile Management	Low	3
Workshift Management	High	6
Book a ride	High	2x6
Ride Allocation	High	6
Taxi Driver Ride Request	Low	3
Total		33

1.1.5 External Outputs

The external outputs shown by the system are all related with the notifications. In fact the system administrators can notify all the users about service situations (for instance a strike, an incident that forbids the access in some city areas and so on). Other kind of notifications, are the ones about the requested ride's status.
Finally we have the operations of shown these notifications (Read the Alerts). All the described operations have to interact with about three types of record (always the user and the alert. If any, also other files are involved, as the ride or the position) and many files (both users or old alerts when we are showing the alerts), thus the complexity is high. Instead, the ride status notifications have to interact with one user, so the complexity is low.

The results are summarized in the following table:

EO	Complexity	FP
System Notifications	High	7
Ride Status Notifications	Low	4
Read The Alerts	High	7
Total		18

1.1.6 Final Results

The Function Point found in the previous sections are reported in the following table:

Function Type	Value
Internal Logic Files (ILF)	31
External Logic Files (ELF)	5
External Inputs (EI)	24
External Inquiries (EQ)	33
External Outputs (EO)	18
Total	111

Since our project has no a specific programming language to be used, in the following table we report the project size estimations both for the C++ and for the Java. In addition we report a few interesting measure with C, Assembler and Machine Code:

Programming Language	UFP to SLOC Conversion Ratios[4]	Lines of Code
Java	53	5833
C++	55	6105
C	128	14208
Assembly - Basic	320	35520
Machine Code	640	71040

[4]The values are still taken from the COCOMO II reference manual.

1.2 Effort Estimation (COCOMO II)