Functional Point and COCOMO

A **function point** is a "unit of measurement" to express the amount of business functionality an information system provides to a user. Function points measure software size. The table below shows the weights values that we have used to calculate the FP value.

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

In order to calculate the FP value we have evaluated these aspects:

1. Internal Logic Files (ILFs)
2. External Logic Files (EIFs)
3. External Inputs (EIs)
4. External Inquiries (EIQs)
5. External Outputs (Eos)

Then we have hypothesized the estimated number of lines of code.

❖ Internal Logic Files

The application stores information about:

- Users
- Drivers
- Requests
- TaxiQueue
- Taxis
- Locations

Locations, taxis and requests are entities that store only few information used by other entities. We can consider them with a simple weight.

Users and Drivers are entities that store a bit of information in more than those before. They often interact with the application. We can consider these two entities with a medium weight.

TaxiQueue is composed by a higher number of locations and automatically manage the drivers within it. We can consider it with a complex weight.

Function Points for ILFs = (3*7)+(2*10)+(1*15) = **56 FPs.**

- **External Logic Files**

The application has to manage the position of each taxis from an external service based on GPS locations. Each of the retrieved position is elaborated by our system in order to establish the right belonging of a driver into a queue.

We can consider this with a complex weight.

Function Points for EIFs = **10 FPs.**

- **External Inputs**

The application interact with users and drivers.

The application allows a user to:

- Login: this is a simple operation related only to the access manager, so we can consider it with a simple weight. **3 FPs.**
- Create a Simple Request: this operation is made on a single form but involve other entities. We can consider it with a medium weight. **4 FPs.**
- Create a Detailed Request: this operation involves not only other entities but also it’s necessary to automatically notify a driver at a specific timing. We can consider it with a complex weight. **6 FPs.**

The application allows a driver to:

- Login: this is a simple operation related only to the access manager, so we can consider it with a simple weight. **3 FPs.**
- Set himself as Available: This involve the insertion of the driver into the queue. This operation can be considered with a medium weight. **4 FPs.**
- Confirm the Requests: Through this operation, a notification is automatically sent to the user’s owner. This operation can be considered with a complex weight. **6 FPs.**

The total amount Function Points for External Input is 3+4+6+3+4+6 = **26 FPs.**

❖ External Inquiries

The application allows a user to view the number of taxi available in his zone according to his phone GPS location. A user can also receive a notification about his accepted request. This can be considered with a simple weight. **3 FPs.**

The application allows a driver to view the pending user's requests in order to confirm them. This can be considered with a medium weight due to the forward system policy. **4 FPs.**

Both users and drivers can see their profile changing eventually some information. This can be considered with a simple weight. 3+3 = **6 FPs.**

❖ External Outputs

There is no external outputs created by the application.

❖ Summarise

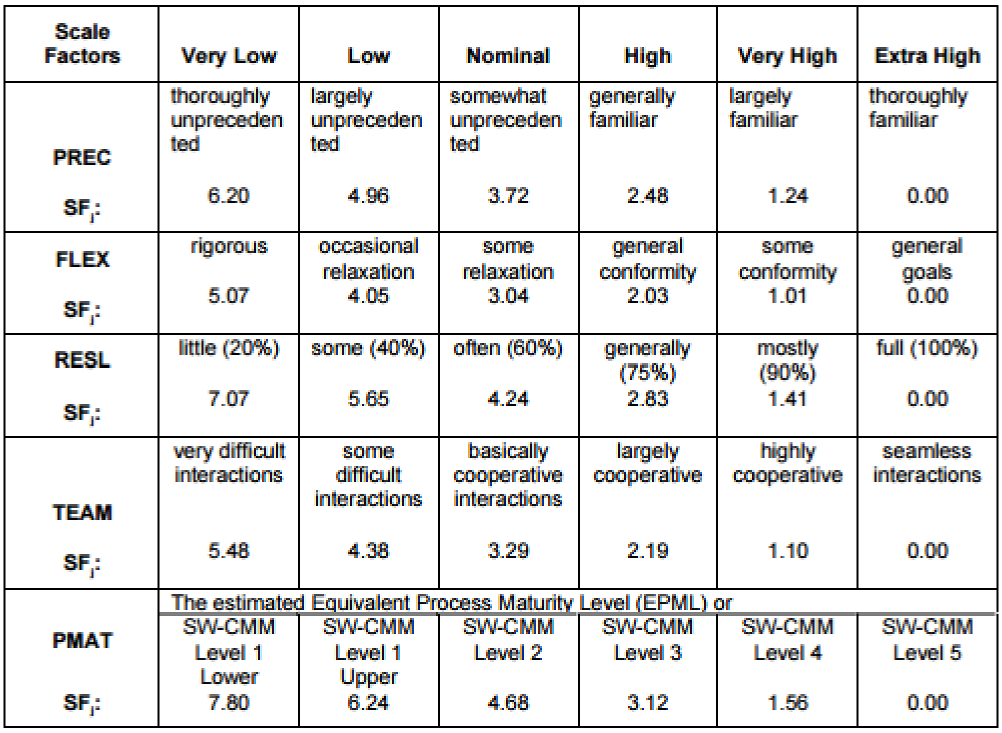
Function Type	*Value*
Internal Logic Files	56
External Logic Files	10
External Inputs	26
External Inquiries	13
External Outputs	-
TOTAL	**105**

Estimation of the size of the project.

With the total amount of Function Points obtained, we can hypostasize the size of the project in terms of lines of code.

LOC = 105 * 46 = 4830 Lines Of Code.

COCOMO II



Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unpreceden ted	largely unpreceden ted	somewhat unpreceden ted	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 – Precedentness**

  It reflects the previous experience in past project like this. For us, this kind of project is the first in our life we are doing and that's why this value will be very low.

- **FLEX – Development flexibility**

  It reflects the degree of flexibility in the development process. The professor left us a large space of flexibility without forcing us with too much details, that's why this value is going to be very high.

- **RESL – Risk resolution**

  <u>TODO</u>

- **TEAM – Team cohesion:**

  It reflects how well the development team know each other and work together. At the beginning of the project we didn't know each other and both of us did not know how the other worked. Although this aspect, we hadn't any problems on work's organization and division of tasks. Due to these considerations, this value will be very high.

- **PMAT – Process maturity:**

  There are two ways of rating Process Maturity. We have chosen the second that s organized around the 18 Key Process Areas (KPAs) in the SEI Capability Maturity Model. We can consider this value as high.

In the table below, we have summarised the values obtained.

Driver factor	*Incidence*	*Value*
Precedentedness	Very low	6.20
Development flexibility	Very high	1.01
Risk resolution	Very high	1.41
Team cohesion	Very high	1.10
Process maturity	High	3.12
	TOTAL	**12.84**

Cost drivers

To evaluating the cost drivers below, we used the table contained in the manual "COCOMO II - Model Definition Manual", available at:

http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf

- **Required Software Reliability**

  This value for us is going to be very high because software failures can be translated with the possibility to being on foot without transport.

- **Database size**
  This cost driver attempts to capture the effect large test data requirements have on product development. Our test database size could be equal to about 250 KB. The estimated program size it's 4370 LOC. The division D/P is 57,20. This parameter is included into the interval 10-100 and that's why the Database size value will be nominal.

- **Product complexity**
  Set to nominal according to the new COCOMO II CPLEX rating scale.

- **Required Reusability**
  This cost driver reflects the reusability of our components. One of our aim was to design the system modular as possible in order to keep this goal. We have identified for each aspect of the program a relative component that manage it. This cost driver value will be high.

- **Documentation match to life-cycle needs:**
  Several software cost models have a cost driver for the level of required documentation. All the documents are already provided for a future development. We can consider a nominal value for this cost driver.

- **Execution time constraint:**
  This is a measure of the execution time constraint imposed upon a software system. The rating is expressed in terms of the percentage of available execution time expected to be used by the system or subsystem consuming the execution time resource. We can consider a very low value for this driver cost.

- **Main storage constraint:**
  This rating represents the degree of main storage constraint imposed on a software system or subsystem. For our system, this aspect is no relevant and we can consider this value as very low.

- **Platform volatility:**
  "Platform" is used here to mean the complex of hardware and software (OS, DBMS, etc.) the software product calls on to perform its tasks. Since we do not have implemented anything yet and the requirements are not going to change frequently, we can consider this value with a low rate.

- **Analyst capability:**
  Analysts are personnel who work on requirements, high-level design and detailed design. Since we have dedicated the entire project on the analysis of requirements and design, we can consider this cost with a very high value.

- **Personnel continuity:**
  Since our available time to do the project was about 4 month, we can consider this value as very low.

- **Application experience:**
  The rating for this cost driver is dependent on the level of applications experience of the project team developing the software system or subsystem. Since it is the first time for us about planning an entire application, we can consider this value as low.

- **Platform experience, Programmer capability, Language and Tool experience:**
  We are not going to consider these values because we didn't have to develop and program anything.

In the table below, we have summarised the values obtained.

Driver factor	*Incidence*	*Value*
Required software reliability	Very high	1.26
Database size	High	1.00
Product complexity	Nominal	1.00
Required reusability	High	1.07
Documentation match to life-cycle needs	Nominal	1.00
Execution time constraint	Very low	n/a
Main storage constraint	Very low	n/a
Platform volatility	Low	0.87
Analyst capability	Very high	0.71
Personnel continuity	Very low	1.29
Application experience	Low	1.10
Platform experience	-	-
Programmer capability	-	-
Language and tool experience	-	-
	PRODUCT	**1.18**

Effort estimation

The final equation gives us the effort estimation measured in Person-Months (PM)

$$\text{Effort} = A * \text{EAF} * \text{KSLOC}^{E}$$

The values of A, B, C, and D in the COCOMO II.2000 calibration are:

A = 2.94

B = 0.91

C = 3.67

D = 0.28

EAF is the product of all the cost drivers that is equal to: 1.18

KSLOC represents the estimated lines of code obtained from the FP analysis: 4830

E is the exponent derived from the Scale Drivers with the equation below:

B+0.01* sum{i} SF[i] = 0.91 + 0.01 * 12.84 = **1.0384**

With all of these parameters we can calculate the final effort:

$\text{Effort} = 2.94*1.18*4.830^{1.0384} = 17.8008$ PM

Schedule estimation

We are going to use this formula to compute the estimated duration:

$$\text{Duration} := 3.67 * \text{Effort}^{F}$$

Where F = 0.28 + 0.2 * (E-B) = 0.28 + 0.2 * (1.0384-0.91) = **0.3057**

So → Duration = $3.67 * 17.8008^{0.3057}$ = **8.84 → 9**

The duration estimated by these computations is not similar to how the reality is. It is also truth that in our project we had just to do the documentation. Probably if we were to do also the implementation and development of the entire application, the duration of the global project could be about 9 months.

P = Effort/ Duration = 17.8008/9 = **1.98 → 2**

Since the estimated number of people for this project is equal to two, and we are two, it is remarkable to think that estimated time found is reasonable.