MeteoCal Project

MeteoCal: Project Reporting

Authors:
Claudio Sanna
Walter Samà

October 2014 - January 2015

Contents

Chapter 1

Function Points Analysis

In order to perform this evaluation we need to know how to find the correct weights for the various functions, so we take for granted that we have to use the default weights table.

Table : FUNCTION COMPONENT COMPLEXITY WEIGHT ASSIGNMENT

Component	Low	Average	High
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6
Internal Logical Files	7	10	15
External Interface Files	5	7	10

Each function will be signed with three different mark **L** (low), **A** (average), **H** (high); which means respectively that the function could be considered Simple, Normal or Complex.

1.1 Internal Logic Files - ILFs

These files represent the internal information that the application have to store regarding:

- User. **L**
- Calendar. **L**
- Event. **A**
- BadConditions. **L**
- Notification. **L**

All these files has a not very complex structure apart from event which has many references to other database elements.
So calculating the ILF: 4 * **L** + 1 * **A** = 4 * 7 + 1 * 10 = **38**.

1.2 External Interface Files - EIFs

These files represent the external information that the application need.

- Weather. **L**

The only file used from an external program is the weather, which is saved in a very simple structure. So the total UFP for the EIFs are = 1 * **L** = 1 * 5 = **5**.

1.3 External Inputs

Services Offered for the users that will use the application, analysing the entities involved in the functions.

- Login: uses only the user table, so it has low weight. **L**
- Register: user user and calendar table, so it still has low weight. **L**
- Event creation: uses more than five tables so it has high weight. **H**
- Event deleting: same complexity as creating event . **H**
- Event updating: same complexity as creating event. **H**
- Change personal settings: involves only user table. **L**
- Change calendar privacy: involves only calendar table. **L**
- Search for user: involves using the user and calendar table. **L**

So the total UFP for the E-inputs = 5 * **L** + 3 * **H** = 5 * 3 + 3 * 6 = **33**.

1.4 External Inquiries

Request of the users to have access to some information stored in the application

- Show Notification: involves using the notification tables. **L**
- Show personal info: involves using the user tables. **L**
- Show calendar: involves using the user and calendar table to know which calendar show, and the event table to show event position on it; so using 3 tables it is of average complexity. **A**
- Show event details: use many tables as weather, badConditions, participant and event; so it uses both ILF and ELF and it is of high complexity. **H**

So the total UFP for the E-inquiries : 2 * **L** + 1 * **A** + 1 * **H** = 2 * 3 + 1 * 4 + 1 * 6 = **16**.

1.5 External Outputs

Output of the application, this is what the application will produce

- Daily weather update: it is realted to weather tables and use the external input; it can still be considered with low complexity. **L**
- One day notification advise: it consider weather, event, badConditions, notification and participant table, as it checks the weather with the bad conditions and create a notification for every users; so it has an high complexity. **H**
- Three day notification advise: it consider weather, event, badConditions, notification and participant table, as it checks the weather with the bad conditions and create a notification for the event creator; so it has an high complexity. **H**
- Create event notification: the others notification can be considered simple as they involve only three tables; so it is an average complexity. **A**

So the total UFP for the E-outputs : 1 * **L** + 1 * **A** + 2 * **H** = 1 * 4 + 1 * 5 + 2 * 7 = **23**.

1.6 Total FP Number and summary

In summary the computed value for the unadjusted FPs is: **115** UFP;
We don't have some historical data that can allow us to compare this value with others, so we can't find the estimated time for the development of the application using this value but we can start from this value using it as a basis to estimate the size of the project In KSLOCs (Kilo Source Lines Of Code), and then use COCOMO II to estimate the effort.

Chapter 2

COCOMOII Analysis

In order to use COCOMOII, we have to know how many KSLOCs will need our project. In order to do this we have to follow some steps.

2.1 Calculate Adjusted Function Point and Source Lines Of Code

In the Function Points Analysis we found the: Total Unadjusted Function Points (UFP) = 115. We could use the UFP to directly calculate the SLOCs, because an adjustment that focus on the functions could not be useful to the predictions of a development costs; but adding the complexity adjustment value, we can consider other aspects of our application to try to find an even better approximation of the Fps, so we will follow this method.
To calculate the Total Complexity Adjustment Value, several factors have to be considered, such as Backup and recovery, code design for reuse, etc. All the factors, called Complexity Weighting Factors, or Degrees of Influence (D_i for simplicity), and their estimated values in this project are shown in the following table. For each Complexity Weighting Factor is assigned a value that ranges between 0 (not important) to 5 (absolutely essential).

Number	Complexity Weighting Factor	Value
1	Backup and recovery	0
2	Data communications	4
3	Distributed processing	1
4	Performance critical	0
5	Existing operating environment	2
6	On-line data entry	1
7	Input transaction over multiple screens	0
8	Master files updated online	0
9	Information domain values complex	1
10	Internal processing complex	5
11	Code designed for reuse	1
12	Conversion/installation in design	2
13	Multiple installations	0
14	Application designed for change	2
	Total complexity adjustment value	19

Then we can find the **Product Complexity Adjustment**:

$$PC = 0.65 + (0.01 * \sum_{i=1}^{14} D_i) = 0.84$$

Note that this value is also called Value Added Factor *VAF*.
Now we can calculate the **Adjusted Function Points**:

$$FP = UFP * PC = 115 * 0.84 = 96.6$$

The last thing that we need to know is the Language Factor, that is the amount of SLOCs per FP. We find this on this site: Function Points Language Table

For Java language, this **Language factor** is assumed as $= 53$

After knowing that we can find the total **Source Lines Of Code**:

$$SLOCs = FP * LF = 96.6 * 53 = 5120$$

And so the

$$KLOCs = SLOCs/1000 = 5.12$$

2.2 COCOMOII

In order to calculate the Effort (measured in Person-Months – PM) we need to know 2 other values:

- the Exponent of the formula - **E**: that is the exponent derived from the *Scale Drivers*.
- the Effort Adjustment Factor - **EAF**: that is the product of the effort multipliers corresponding to each of the cost drivers for our project.

We search for some Driver Values on the net and we find this table in the document that we found here Calibrating the COCOMO II Post-Architecture Model - University of Southern California, Los Angeles

Table 2. Apriori Model Values

Driver	Sym	VL	L	N	H	VH	XH
PREC	SF_1	0.05	0.04	0.03	0.02	0.01	0.0
FLEX	SF_2	0.05	0.04	0.03	0.02	0.01	0.0
RESL	SF_3	0.05	0.04	0.03	0.02	0.01	0.0
TEAM	SF_4	0.05	0.04	0.03	0.02	0.01	0.0
PMAT	SF_5	0.05	0.04	0.03	0.02	0.01	0.0
RELY	EM_1	0.75	0.88	1.00	1.15	1.40	
DATA	EM_2		0.94	1.00	1.08	1.16	
CPLX	EM_3	0.75	0.88	1.00	1.15	1.30	1.65
RUSE	EM_4		0.89	1.00	1.16	1.34	1.56
DOCU	EM_5	0.85	0.93	1.00	1.08	1.17	
TIME	EM_6			1.00	1.11	1.30	1.66
STOR	EM_7			1.00	1.06	1.21	1.56
PVOL	EM_8		0.87	1.00	1.15	1.30	
ACAP	EM_9	1.5	1.22	1.00	0.83	0.67	
PCAP	EM_{10}	1.37	1.16	1.00	0.87	0.74	
PCON	EM_{11}	1.26	1.11	1.00	0.91	0.83	
AEXP	EM_{12}	1.23	1.10	1.00	0.88	0.80	
PEXP	EM_{13}	1.26	1.12	1.00	0.88	0.80	
LTEX	EM_{14}	1.24	1.11	1.00	0.9	0.82	
TOOL	EM_{15}	1.20	1.10	1.00	0.88	0.75	
SITE	EM_{16}	1.24	1.10	1.00	0.92	0.85	0.79
SCED	EM_{17}	1.23	1.08	1.00	1.04	1.10	

Figure 2.1: Driver Values

With these values we calculate the drivers' value for our project choosing from the right categories.

2.2.1 Exponent value Calculation

Scale Driver	Complete Name	Category	Value
PREC	Precedentedness	L	0.04
FLEX	Development Flexibility	L	0.04
RESL	Risk Resolution	VL	0.05
TEAM	Team Cohesion	VH	0.01
PMAT	Process Maturity	N	0.03

Then following the method to calculate the exponent E that we found in the previous document, we find that:

$$E = 1.01 + \sum_{i=1}^{5} V_i = 1.01 + 0.17 = 1.18$$

where V_i is the value of the i-th Scale Driver.

2.2.2 Effort Adjustment Function Calculation

Cost Driver	Complete Name	Category	Value
RELY	Required Software Reliability	L	0.88
DATA	Data Base Size	L	0.94
CPLEX	Product Complexity	L	0.88
RUSE	Required Reusability	L	0.89
DOCU	Documentation match	N	1
TIME	Execution Time Constraint	N	1
STOR	Main Storage Constraint	N	1
PVOL	Platform Volatility	L	0.87
ACAP	Analyst Capability	N	1
PCAP	Programmer Capability	N	1
PCON	Personnel Continuity	H	0.91
AEXP	Applications Experience	VL	1.23
PEXP	Platform Experience	VL	1.26
LTEX	Language and Tool Experience	L	1.11
TOOL	Use of Software Tools	N	1
SITE	Multisite Development	N	1
SCED	Required Development Schedule	H	1.04

Then following the method to calculate the EAF, we find that:

$$EAF = \prod_{i=1}^{17} V_i = 1.01 + 0.17 = 0.9176$$

where V_i is the value of the i-th Scale Driver.

2.2.3 Effort Calculation

Finally the Effort is calculated:

$$Effort = 2.94 * EAF * (KSLOC)^{E} = 2.94 * 0.9176 * (5.12)^{1.18} = 18.53 Person - Month$$

Considering that out group is formed by 2 People and that $NPeople = Effort/Duration$ we could obtain the theoretical duration:

$$Duration = Effort/Npeople = 9.25 Month$$

But in the reality we have been working on this project since the end of october so 4 months have passed, if we try to recalculate the effort by multiplying the Npeople and the Effective Duration we will obtain:

$$Effort = NPeople * Duration = 2 * 4 = 8PM$$

.

2.3 COCOMOII Using a Tool

After some research on the net, we found an interesting tool for the calculation of COCOMOII CO-COMOII Constructive Cost Model Tool this tool can use as input the UFP or the SLOCs, and the cost drivers categories so we tried to insert the same parameters that we calculate manually and the result are these:

2.3.1 Effort Using UFP

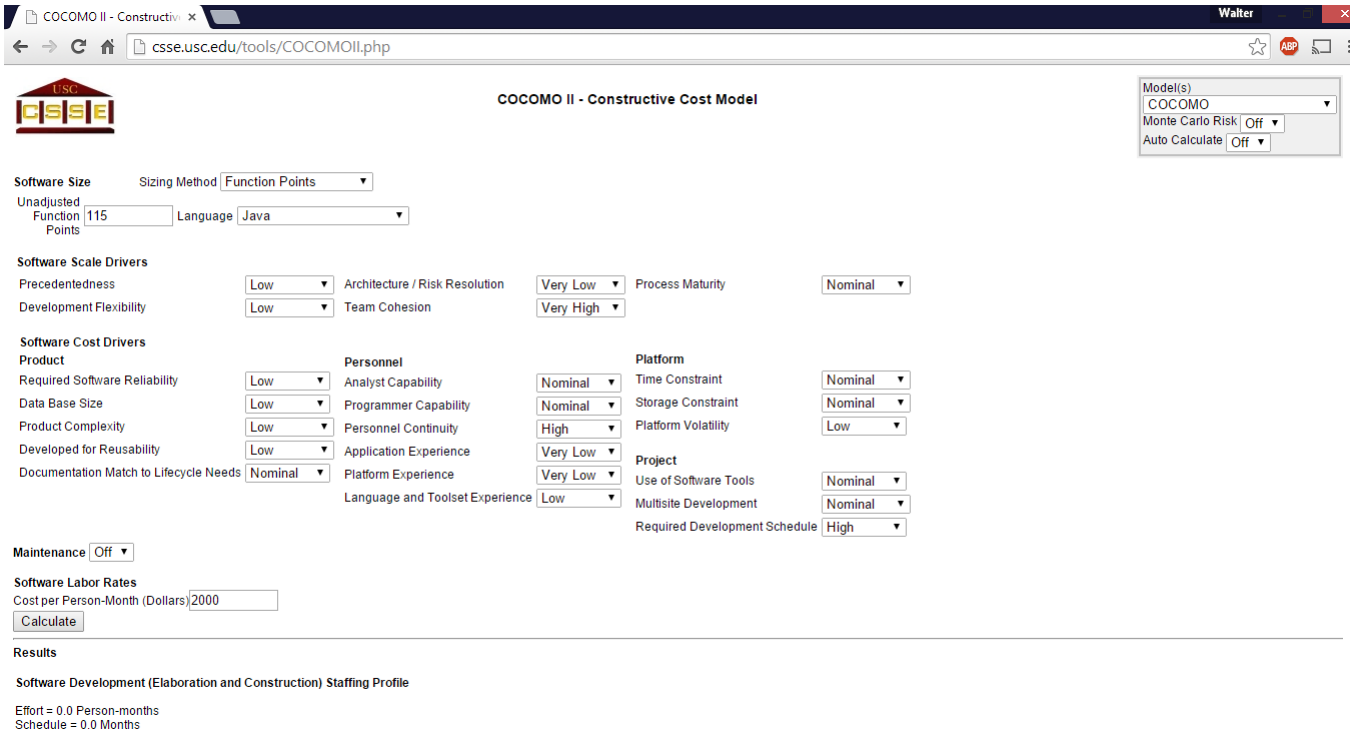
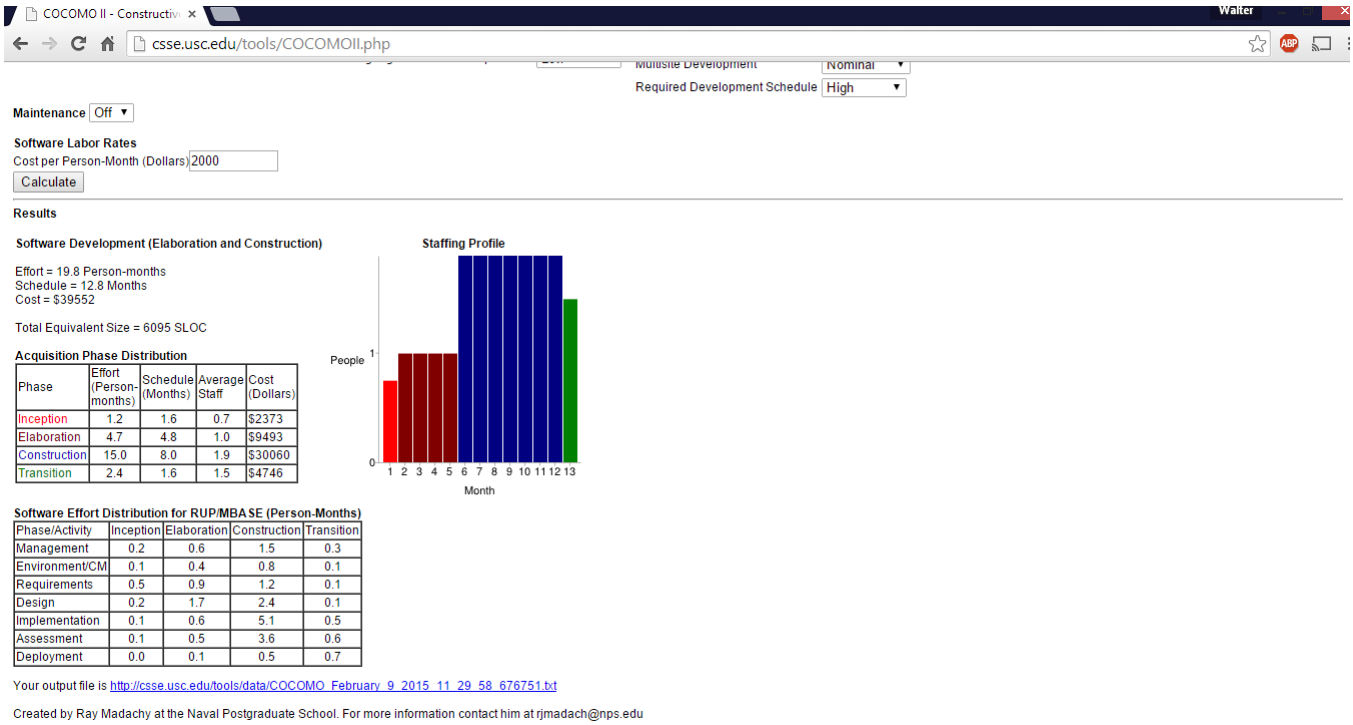


Figure 2.2: COCOMOII Tool UFP Parameters



Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	1.2	1.6	0.7	$2373
Elaboration	4.7	4.8	1.0	$9493
Construction	15.0	8.0	1.9	$30060
Transition	2.4	1.6	1.5	$4746

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.2	0.6	1.5	0.3
Environment/CM	0.1	0.4	0.8	0.1
Requirements	0.5	0.9	1.2	0.1
Design	0.2	1.7	2.4	0.1
Implementation	0.1	0.6	5.1	0.5
Assessment	0.1	0.5	3.6	0.6
Deployment	0.0	0.1	0.5	0.7

Figure 2.3: COCOMOII Tool UFP Result

2.3.2 Effort Using SLOCs

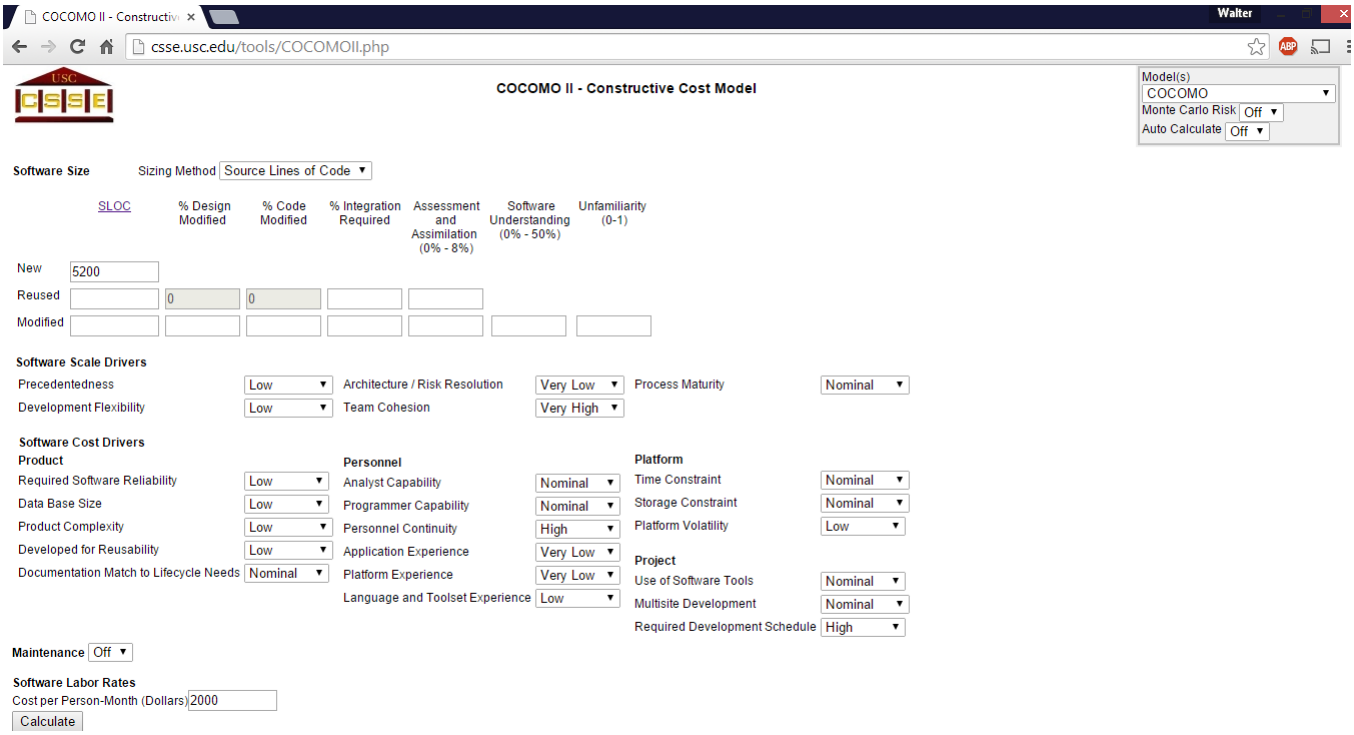
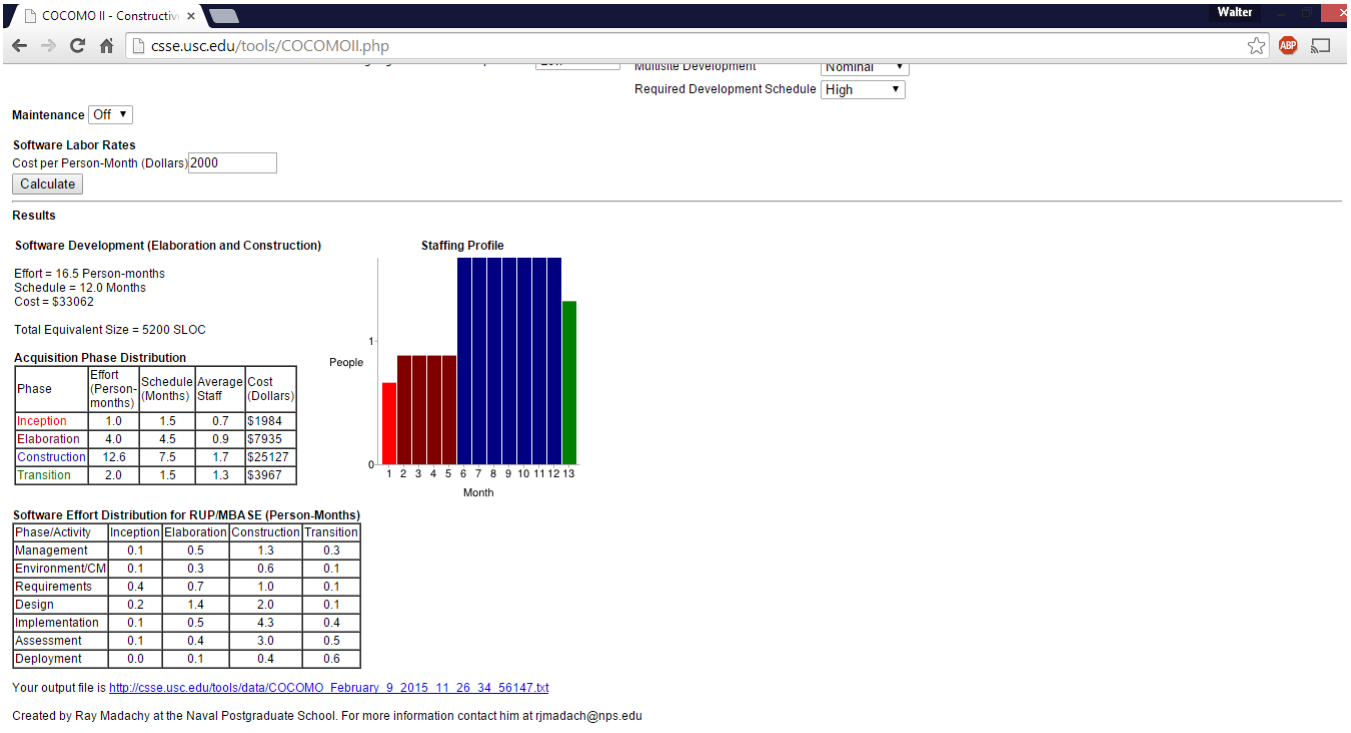


Figure 2.4: COCOMOII Tool SLOCs Parameters



Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	1.0	1.5	0.7	$1984
Elaboration	4.0	4.5	0.9	$7935
Construction	12.6	7.5	1.7	$25127
Transition	2.0	1.5	1.3	$3967

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.1	0.5	1.3	0.3
Environment/CM	0.1	0.3	0.6	0.1
Requirements	0.4	0.7	1.0	0.1
Design	0.2	1.4	2.0	0.1
Implementation	0.1	0.5	4.3	0.4
Assessment	0.1	0.4	3.0	0.5
Deployment	0.0	0.1	0.4	0.6

Figure 2.5: COCOMOII Tool SLOCs Result

2.4 Final Conclusion and Consideration

Obviously we find an upper bound of the real Effort for the development of our application, but looking at the result obtained using the tool and the one we obtained using the default tables that we found; we can say to have done a good job because the results are similar. The only thing we could say is that we can't know what is difficult for us and what is objectively difficult, so maybe the number of unadjusted function points could be less than the number we found; but considering that this is an estimation utilities that have to find an upper bound for the development of an application the Effort we found could be right.

Bibliography

[1] Cocomo ii - constructive cost model tool. http://csse.usc.edu/tools/COCOMOII.php.

[2] Cocomo ii cost driver and scale driver help. http://sunset.usc.edu/research/COCOMOII/expert_cocomo/drivers.html.

[3] Calculating function points. http://www.codeproject.com/Articles/18024/Calculating-Function-Points, 2007.

[4] Calibrating the cocomo ii post-architecture model. http://sunset.usc.edu/csse/TECHRPTS/1998/usccse98-502/CalPostArch.pdf, 2015.

[5] Function points language table. http://www.qsm.com/resources/function-point-languages-table, 2015.

[6] Andrew Damian Tamburri. Cost estimation slides, 2014.