Politecnico di Milano

A.A. 2014-2015

Software Engineering 2: “MeteoCal” **P**roject **R**eporting

Federico Migliavacca (mat. 836582), Leonardo Orsello (mat. 837176)

10 February 2015

Contents

1 Introduction

In this document the time and resources necessary to the the development of the MeteoCal application are evaluated. For the analisys the Functional Point approach has been used, and the results have been compared with the dimension of the project. Furthermore, the COCOMO model has been used to evaluate the effort for MeteoCal implemantetion and the results have been compared whit the time actually spent.

2 Function Point Approach

The Functional Point approach is a technique that allows to evaluate the effort needed for the design and implementation of a project. We have used this technique to evaluate the application dimension basing on the funcionalities of the aplication itself. The funcionalities list has been obtained from the RASD documet and for each one of them the realization complessity has been evaluated. The functionalities has been groped in:

- **Internal Logic File:** it represents a set of homogeneous data handled by the system. In MeteoCal application this corrisponds to the database table.
- **External Interface File:** it represents a set of homogeneous data used by the application but handled by external application. In MeteoCal application there are no such functions.
- **External Imput:** elementary operation that allows input of data in the system.
- **External Output:** elementary operation that creates a bitstream towards the outside of the application.
- **External Inquiry:** elementary operation that involves input and output operations.

The following table outline the number of Functional Point based on funtionality and relative complexity:

Function Type	Complexity		
	Simple	Medium	Complex
Internal Logic File	7	10	15
External Interface File	5	7	10
External Input	3	4	6
External Output	4	5	7
External Inquiry	3	4	6

Using the Function Points method, the estimation of the effort required to complete the application depends on the functionalities the application has to offer. Specifically, the number of FPs can be computed as the weighted sum of function types using the coefficient of the table above. We perform the calculation step by step:

- **Internal Logic Files (ILFs):** The application includes a number of ILFs that will be used to store the information about users, calendar, events, weather forecast and notification. Each of these entities has a simple structure as it is composed of a small number of fields except for events entity that have a more complex structure. Thus, we can decide to adopt medium weight for event and simple weight for all other entities. This means that we will come out with 4 x 7 + 1 x 10= 38 FPs concerning ILFs.
- **External Logic Files (ELFs):** The application features also only one EIFs to manage the interaction weather APIs. This information results in only one entity with a simple structure. Thus, we decide to adopt a simple weight. As a result, we get 1 x 5 = 5 FPs.
- **External Inputs:** The application interacts with the user to allow him/her to:
  - *Login/logout:* these are simple operations, so we can adopt the simple weight for them. 2 x 3 = 6 FPs
  - *Became a registered user:* this is a simple operation, so we can adopt the simple weight. 1 x 3 = 3 FPs
  - *Create a new event in the calendar:* this is not a simple operation because it involves four entities: user, event, calendar, weather forecast; so we can adopt a complex weight. 1 x 6 = 6 FPs
  - *Modify an existing event:* this is not a simple operation because it involves four entities: user, event, calendar, weather forecast; so we can adopt a complex weight. 1 x 6 = 6 FPs
  - *Move and resize an existing event:* this is not a simple operation because it involves four entities: user, event, calendar, weather forecast; so we can adopt a complex weight. 1 x 6 = 6 FPs
  - *Delete an existing event:* this operation involves four entities: user, event, calendar and weather forecast; but it is simple so we can adopt the medium weight. 1 x 4 = 4 FPs
  - *Invite/delete other user to a specific event of his/her calendar:* these are simple operations, so we can adopt the simple weight. 2 x 3 = 6 FPs
  - *See detail and weather forecast of a specific event of his/her calendar:* this operation involves three entities: user, event, calendar; but it is a simple operation, so we can adopt the simple weight. 1 x 3 = 3 FPs

- **External Output:**
  - *After login, application will notify only the creator user three days before an event takes place if the weather is not good:* this operation involves five entities, user, event, calendar, notify and weather forecast; this is quite complex so we can adopt complex cost. 1 x 7 = 7 FPs
  - *After login, application will notify invited user one days before an event takes place if the weather is not good:* this operation involves five entities; user, event, calendar, notify and weather forecast; this is quite complex so, again, we can adopt complex cost. 1 x 7 = 7 FPs
  - *After login, application will notify to the user that he/she has been invited to an event and leaves to accept or decline invitation:* this operation involves in five entities: user, event, calendar, notify and weather forecast; this is quite complex so we can adopt complex cost. 1 x 7 = 7 FPs
- **External Inquiries:** The application allows user to request information events, in particular user can see:
  - *The public event of other registered user:* this operation involves three entities: user, event, calendar; but it is a simple operation, so we can adopt the medium weight. 1 x 4 = 4 FPs
  - *Event of public calendar of other registered user:* this operation involves three entities: user, event, calendar; but it is a simple operation, so we can adopt the medium weight. 1 x 4 = 4 FPs
  - *Event to which has been invited:* this operation involves three entities: user, event, calendar; but it is a simple operation, so we can adopt the medium weight. 1 x 4 = 4 FPs
- **Total FP number and summary:** In summary, we have computed the following value for the unadjusted FPs: 116. This value can be used directly to estimate the effort in case we have some historical data that tell us how much time we usually take for developing a FP. Otherwise, it can be used as a basis to estimate the size of the project in KLOC and then use another approach such as COCOMO to estimate the effort.

3 COCOMO Approach

To pass from FP to SLOC we use an average conversion factor of 46 as described at http://www.qsm.com/resources/function-point-languages-table, an updated version that adds J2EE of the table included in official manual (http://sunset.usc.edu/research/COCOMOII/Docs/modelman.pdf).

$$116\,FPs * 46 = 5336\,SLOC$$

A first estimation is based on FPs approach converted to SLOC. We Consider a project with all "Nominal" Cost Drivers and Scale Drivers would have an EAF of 1.00 and exponent E of 1.0997. Following this formula

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

we obtain

$$effort = 2.94 * (1.0) * (5.336)^{1.0997} = 18.53\,Person/Months$$

EAF: Effort Adjustment Factor derived from Cost Drivers.
E:Exponent derived from Scale Drivers.
Now we try to calculate the schedule (duration) of project in month with the following formula

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

We consider an exponent E of 0.3179

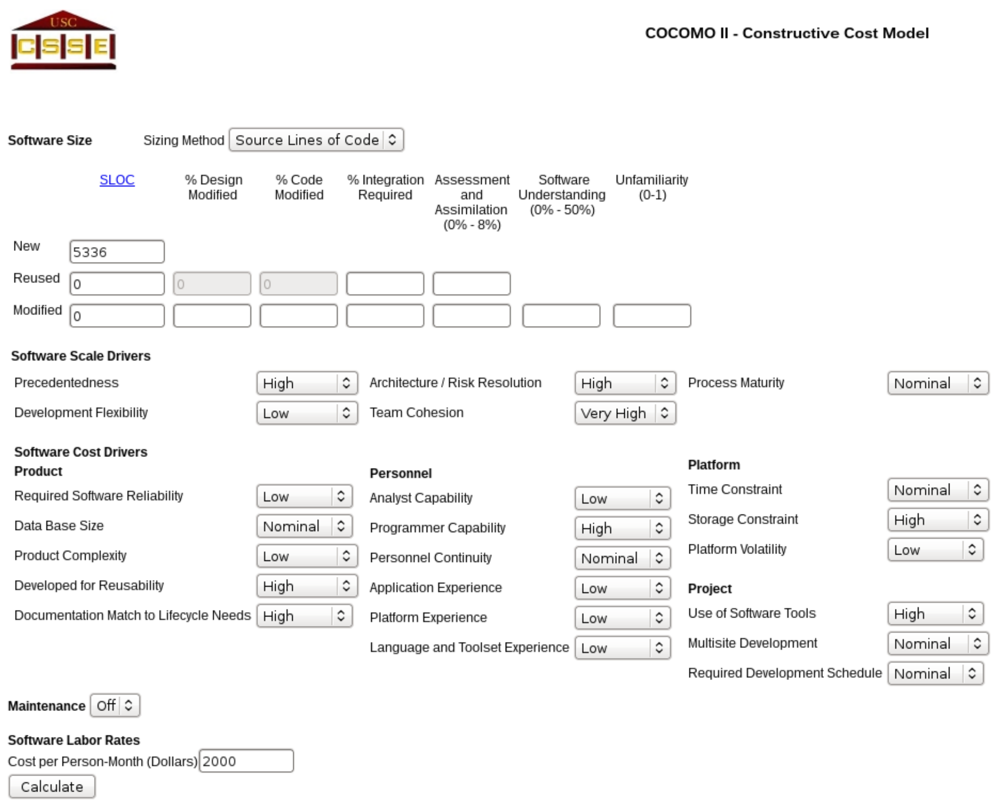
$$Duration = 3.67 * (18.53)^{0.3179} = 9.28\,Months$$

Now we can estimate the number of people needed to complete the project with the following formula

$$N_{people} = effort/Duration$$

$$N_{people} = 18.53/9.28 = 2\,people$$

We want to give a more precise estimation adjusting some Scale Driver. To evaluate the COCOMO II and determine the effort required to complete the software project we also use an online tool that helps us to do some calculus (http://csse.usc.edu/tools/COCOMOII.php). We add the report of that site and the choice made about the Scale Driver to obtain that result.



USC CSSE

COCOMO II - Constructive Cost Model

Software Size Sizing Method: Source Lines of Code

	SLOC	% Design Modified	% Code Modified	% Integration Required	Assessment and Assimilation (0% - 8%)	Software Understanding (0% - 50%)	Unfamiliarity (0-1)
New	5336						
Reused	0	0	0				
Modified	0						

Software Scale Drivers

Precedentedness	High	Architecture / Risk Resolution	High	Process Maturity	Nominal
Development Flexibility	Low	Team Cohesion	Very High		

Software Cost Drivers

Product

Required Software Reliability	Low
Data Base Size	Nominal
Product Complexity	Low
Developed for Reusability	High
Documentation Match to Lifecycle Needs	High

Personnel

Analyst Capability	Low
Programmer Capability	High
Personnel Continuity	Nominal
Application Experience	Low
Platform Experience	Low
Language and Toolset Experience	Low

Platform

Time Constraint	Nominal
Storage Constraint	High
Platform Volatility	Low

Project

Use of Software Tools	High
Multisite Development	Nominal
Required Development Schedule	Nominal

Maintenance Off

Software Labor Rates

Cost per Person-Month (Dollars) 2000

Calculate

Results

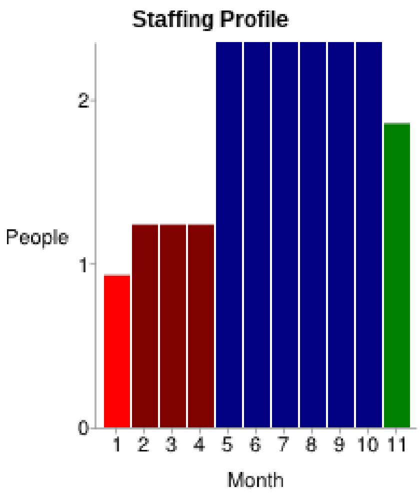
Software Development (Elaboration and Construction)

Effort = 18.6 Person-months
Schedule = 9.6 Months
Cost = $37194

Total Equivalent Size = 5336 SLOC

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	1.1	1.2	0.9	$2232
Elaboration	4.5	3.6	1.2	$8927
Construction	14.1	6.0	2.3	$28268
Transition	2.2	1.2	1.9	$4463



Software Effort Distribution for RUP/MBASE (Person-Months)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.2	0.5	1.4	0.3
Environment/CM	0.1	0.4	0.7	0.1
Requirements	0.4	0.8	1.1	0.1
Design	0.2	1.6	2.3	0.1
Implementation	0.1	0.6	4.8	0.4
Assessment	0.1	0.4	3.4	0.5
Deployment	0.0	0.1	0.4	0.7

This estimation results not very different from the previous with all “Nominal” Cost Drivers. Probably some parametres compensate other ones and the estimation result is very similar.

4 Conclusion

Here are listed the real hours of works, included in Development Time Document, spent for MeteoCal project.

- **R**equirements **A**nalysis and **S**pecifications **D**ocument:
  - Federico Migliavacca: ~37 hours.
  - Leonardo Orsello: ~37 hours.
- **D**esign **D**ocument
  - Federico Migliavacca: ~27 hours.
  - Leonardo Orsello: ~27 hours.
- **I**mplementation and **T**esting
  - Federico Migliavacca: ~117 hours.
  - Leonardo Orsello: ~117 hours.
- **A**cceptance **T**esting
  - Federico Migliavacca: ~7 hours.
  - Leonardo Orsello: ~7 hours.
- **F**inal **P**resentation
  - Federico Migliavacca: ~4 hours.
  - Leonardo Orsello: ~4 hours.

The total hours of work spent during all phases of the project are 384 hours.

$$384\, hours/(40 * 4)\, hours = 2.4\, Person/Months$$

we suppose that a man can work 40 hours in a week so 40*4 is a number of hours that man works in a month.
Here is a comparative table between the estimated value and real value.

	Estimated Value	Effective Value
Effort	18.53person/months	2.4 person/months
Duration	9.28 months	4 months
Number of People	2	2
SLOC	5336	2048

We can obviously notice that the estimation of COCOMO II is oversized respect to the real time spent for the project. This could be linked to the statistic nature of COCOMO II itself or the Funcion Point estimation is clearly exagerate regard the real line of code. Another possibility is that in the 4 month of work we have spent more and more time respect of what we declared in the document probably because during night passed with Glassfish we lost the real perception of time.