Politecnico di Milano

MeteoCal

Project Reporting Document

Authors:

DAVERIO Matteo

DE MARIA Valerio

February 9, 2015

Contents

1. Introduction

In this document we analyze our project using two different approaches. First, we try to use the Function Point approach to check If the result is similar to the actual size of the project. In the second part we apply COCOMO formulas and compare it with the real effort spent to realize the project.

In order to do this, we provide some useful data, like Lines of Code(LOC) and hours of work.

COCOMO II provides an estimation of the overall effort needed to develop the project. We want to make a comparison between the estimation and the actual time needed by the team to accomplish the task. In order to do so we kept track of the time each member of the team spent on the project.

1. T_{RASD} = 20h + 20h = 40h

2. T_{DD} = 20h + 20h = 40h

3. T_{DEV} = 140h + 140h = 280h

The overall time spent by the two members of the team on the project turns out to be:

$T_{TOT} = T_{RASD} + T_{DD} + T_{DEV}$ = 40h + 40h + 200h = 360h

In order to compute the number of Lines of Code (LOC) we used a plugin for NetBeans called Simple Code Metrics. Using this tool we computed an actual size of 6717 LOC. This number does not take into account the xhtml pages. Obviously we didn't take into consideration the lines of comments and testdrivers.

2. Function Points

In order to make a comparison between lines of code, we need first to find out the function points(FP).

1. Internal Logical Files (ILF)

a) The system has to store information about the user and his events.
Weight: medium
User's table has a lot of field, and can be associated to an high number of events. Also, system has to manage security settings like password encryption.
b) The system has to store events.
Weight: medium
Events are composed by several fields.
c) The system has to store invitations.
Weight: simple
Invitations have a small number of parameters.
d) The system has to store notifications.
Weight: simple
Notifications have a small number of parameters.

2. External Interface Files (EIF)

a) The system has to retrieve and store forecast's data
Weight: complex
The weight is complex because the management of external forecasts' data requires parsing information coming from the external service.

3. External Inputs (EI)

a) Login/Logout
Weight: simple
Nothing complex, simply control if the input field are correct.
b) Registration
Weight: simple
Checks on input values, inserting new values in a tuple in the database.
c) Change calendar's privacy
Weight: simple
These operation consists only in updating a tuple in the user table.
d) Event creation, modification
Weight: complex

These operations involves the insert/update of a tuple in the event table. When it happens, it causes to create a notification to other users.

e) Search users
Weight: simple
System has to search with a query to the database his data.

f) Answer to notifications
Weight: complex
After the user answers a notification the system has to take care of the update of several different tables.

4. External Inquiries (EIQ)

a) Browse personal calendar
Weight: simple
Show all the events in which users take part.

b) Browse external calendars
Weight: medium
The system has to take care of privacy settings for both users' calendar and their events. It has to display the "external" calendar and its events according to the user's settings.

5. External Outputs (EO)

a) Display weather forecasts
Weight: complex
The system has to manage the interaction with the external service providing the forecasts.

b) Display notifications
Weight: complex
System has to find several cases, and notify them. It access some database's tables.

c) Display invitations
Weight: complex
System has to find invitations to events. It access some database's tables.

To estimate the LOC we will use the following table of weight:

Function Types	**Weights**		
	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

To calculate the Unadjusted Function Point(UFP) we use these contributes:

- **ILF** = 2 * *simple* + 2 * *medium* = 34
- **EIF** = 1 * *complex* = 10
- **EI** = 4 * *simple* + 2 * *complex* = 28
- **EIQ** = 1 * *simple* + 1 * *medium* = 7
- **EO** = 3 * *complex* = 21

UFP = 34 + 10 + 28 + 7 + 21 = 100 FPs

To obtain the estimation of the LOC, we have to use the formula:

LOC = AVG * UFP

For AVG, we use the coefficient specific for J2EE, which is 46

LOC = 46 * 100 = 4600

3. Cocomo II

Cocomo II is useful to estimate the effort needed for a project, measured in Person-Months. For this part we use the Cocomo II manual at (http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf).

In order to do so, we show the general formula, then we go on try to find the values needed:

$$EFFORT = A * Size^{E} * \prod_{i=1}^{n} EM_i$$

Where A=2,94 and Size expressed in KSLOC. Exponent E and Effort Multiplier EM have to be calculated.

1. Exponent

The exponent is obtained as follows:

$$E = B + 0.01 * \sum_{i=1}^{5} SF_i$$

Where B = 0,91.

SF are scale factors, which represent economies or diseconomies of scale encountered.

To estimate SF we use the following table of weight, where each scale factor have a different weight in relation to a rating level, from Very Low to Extra High.

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

This is the list of the scale factors rating levels we estimate for our project:

- **Precedentedness (PREC)**

  Rating level: *low*

  We never developed with J2EE environment and web-based application, but we has one previous experience with java language.

- **Development Flexibility (FLEX)**

  Rating Level: *high*

  The only constraints we had were on the language to be used and on functional requirement that the system had to fulfill.

- **Architecture / Risk Resolution (RESL)**

  Rating level: *nominal*

  We try to spend lot of time on project system architecture, but it was very difficult and not accurate because we don't have any previous experience.

- **Team Cohesion (TEAM)**

  Rating level: *high*

  We have already work on a project, so we have some experience on work in group.

- **Process Maturity (PMAT)**

  Rating level: *nominal*

  We chose level 2 because processes have been planned and documented at the project level.

Having chosen the rating levels for each scale factor we can now calculate the exponent:

$$E = 0.91 + 0.01 * (4.96 + 2.03 + 4.24 + 2.19 + 4.68) = 1.091$$

The exponent shows that economies and diseconomies of scales are almost in balance. This is mainly due to the fact that, despite the lack of previous experience, the team was small and easy to organize.

2. Effort Multiplier

The seventeen effort multipliers (EM) are used in the COCOMO II model to adjust the nominal effort, Person-Months, to reflect the software product under development. Each multiplicative cost driver is defined below by a rating level and a corresponding effort multiplier (the values are taken from the document referenced at the beginning of this part).

- **Required Software Reliability (RELY)**
  Rating level: low
  Effort multiplier: 0.92
  Software failures give rise to easily recoverable losses. There are neither financial losses nor risk for human life.
- **Data Base Size (DATA)**
  Rating level: low
  Effort multiplier: 0,9
  We tested the system without the need of huge testing data sets.
- **Product Complexity (CPLX)**
  Rating level: low
  Effort multiplier: 0,87
  Considering all the different parts of the system (including the gui and the data management system) the overall complexity was modest if compared with industry standards.
- **Developed for Reusability (RUSE)**
  Rating level: nominal
  Effort multiplier: 1,00
  The reusability focus was across project. We designed project's components to be easily reusable.
- **Documentation match to lifecycle needs (DOCU)**
  Rating level: nominal
  Effort multiplier: 1,00
  The documentation covers the basic needs of the project. The requirements analysis and the design document are complete and detailed.
- **Execution Time Constraint (TIME)**
  Rating level: nominal
  Effort multiplier: 1,00
  The system didn't require high execution time to perform complex tasks.
- **Main Storage Constraint (STOR)**
  Rating level: nominal
  Effort multiplier: 1,00
  The system used less (way less) of the 50% of available storage space.
- **Platform Volatility (PVOL)**
  Rating level: low
  Effort multiplier: 0,87
  The development environment and the used tools didn't change significantly during our work.

- **Analyst Capability (ACAP)**
  Rating level: high
  Effort multiplier: 0,85
  The team had good design ability and was able to efficiently cooperate in the analysis of the project.
- **Programmer Capability (PCAP)**
  Rating level: high
  Effort multiplier: 0,88
  The team was able to efficiently make use of members' programming skills with a good organization of the work.
- **Personnel Continuity (PCON)**
  Rating level: very high
  Effort multiplier: 0,81
  The team didn't undergo any personnel turnover.
- **Applications Experience (APEX)**
  Rating level: very low
  Effort multiplier: 1,22
  The team experience in developing applications of this kind was null, we learn everything on this environment during the coding part of the project.
- **Platform Experience (PLEX)**
  Rating level: very low
  Effort multiplier: 1,19
  The considered platform was completely new to the team.
- **Language and Tool Experience (LTEX)**
  Rating level: low
  Effort multiplier: 1,09
  The programming language that has been mainly used is Java. Both team members have an experience with this programming language that is less then a year.
- **Use of Software Tools (TOOL)**
  Rating level: nominal
  Effort multiplier: 1,00
  The team utilized basic lifecycle tools with which was already experienced (we used similar tools in the Software Engineering 1 course the last year).
- **Multisite Development (SITE)**
  Rating level: low
  Effort multiplier: 1,09
  The team members live in different cities and, during the project, had to communicate using individual phones or Social Networks.
- **Required Development Schedule (SCED)**
  Rating level: nominal
  Effort multiplier: 1,00

The time scheduled for planning activities has been fully used in order to develop the RASD and the Design Document.

We define the Effort Adjustment Factor (EAF) as the product of the effort multipliers corresponding to each of the cost drivers for the project.

Given the previous cost drivers analysis we can calculate it:

EAF = 0.92∗0.9∗0.87∗1∗1∗1∗1∗0.87∗0.85∗0.88∗0.81∗1.22∗1.19∗1.09∗1∗1.09∗1 = 0.655

3. Effort Computation

Now we have all the data needed to compute the effort.

KSLOC = SLOC/1000 = 6717/1000 = 6.717; A = 2.94; E = 1.091; EAF = 0.655

The effort is:

$$PM = A * EAF * (KSLOC)^E = 2.94 * 0.655 * (6.717)^{1.091} \approx 15.38$$

4. Final Consideration

COCOMO II treats the number of person-hours per person-month, PH/PM, as an adjustable factor with a nominal value of 152 hours per Person-Month. This number excludes time typically devoted to holidays, vacations, and weekend time off.

If we compute the number of person-hours corresponding to the computed effort we obtain:

$$PH = 152 * 15.38 = 2338(person - hours)$$

The time the team actually devoted to the project is 360 hours. The estimation is bigger then real time. This may be due to the fact that COCOMO II has been developed to deal with big projects and teams. Another reason could be that this is the time needed to provide a project without bugs, with optimized code and with advanced GUI. Obviously we can't reach that goal because we don't have enough time to do it and maybe this is the principal reason we don't meet the Cocomo II estimated time.

HOURS SPENT:

Matteo Daverio: Document creation => (6h).