Politecnico di Milano

Software Engineering 2

MeteoCal

Project reporting

Authors:
Andrea Enrico Turri
Andrea Salmoiraghi
Fabiano Riccardi

10/2/2015

Contents

1 Introduction

In this document we show the estimation of UFP to calculate the number of lines of code that will be compared with the real number written in the implementation. Then, we will explain the computation of COCOMO II parameters and how we obtain the prevision about the effort required from our project.

1.1 Notation

Here the notation used in this document.

EM	Effort Multiplier
FP	Function Point
PM	Personnel-Months
SF	Scale Factor
SLOC	Source Line of Code
TDEV	Time of Developing
UFP	Unadjusted Function Point

2 Function Points

We use this table to evaluate the weight for every function:

FUNCTION TYPES	**WEIGHTS**		
	Simple	Medium	Complex
N. Inputs	3	4	6
N. Outputs	4	5	7
N. Inquiry	3	4	6
N. Internal Files	7	10	15
N. External Files	5	7	10

Now we explain every single choice:

External Input

- Login/logout: *simple*. It require only email and password.
- Registration: *simple*. More field than login but anyway simple.
- Create and updating an event: *complex*. More filed, check and it work on 3 entity.
- Deletion an event: *simple 3*. Remember that sends email to the participant, but this is a simple function.
- Changing settings: *medium*. The same field than registration plus profile image.
- Searching (user or event): *simple*. Only an input box.
- Retrieve password: *simple*.
- Import: *complex*. It have to do a lot of control.

External Output Things that are created from the system for the user:

- retrieve password: *simple*. Creation of the code to allow the changing of the password.
- export: *medium*. Create a file with the entire user's calendar.

External Enquiries Actions that required data retrieval from db:

- Create and updating an event: *medium*. Check overlap and other small thing.
- Calendar view: *complex*. It must show a lot of events.
- Changing settings and showing user: *simple*. This 2 op concern the user so we could consider only one.
- Searching (user or event): *medium*.

Internal Logic File Entity of the system:
We have user, event and notification and meteo all them are *simple* structure: 7*4. They are table with a different number of columns, but nothing will be very large.

External Logic Files Comunication between different software:

- When create and updating an event the system retrieves the meteo from an external website: *medium*.
- Import: *complex*. The file could be very large.

We have to sum all the value to obtain the total amount of UFP.The math tell us that the sum is 112. Now we can estimate the SLOC with a simple multiplication:

$$SLOC = K \times UFP = 29 \times 112 = 3248$$

Where K is a constant dependent on the program language (Java). In the table we didn't find Java, but we found C++. We suppose that Java is similar as LOC, so we chose K=29.

3 Cocomo

Now we are ready to estimate the effort (expressed in month-person), the number of person required and the total time. We could start from Scaling Drivers: these are 5 parameter that modify exponentially the final size. The possibile value are for each driver are: *Very low*, *Low*, *Nominal*, *High*, *Very high*, *Extreme high*. Every value have a different corresponding number dependent on the current scale factor. Let's see the detail:

1. Precedentedness (PREC): *very high*. All the team have tried before to work with Java and with web-developing, so we believe that we have good awareness about this project.

2. Develop Flexibility(FLEX): *high*. We established the requirement and we can only add feature. No particular external interface and no premium for early completion.

3. Architecture/Risk Resolution (RESL): *very high*. We software haven't critical point, we have top-level libraries like JSF and Primefaces and we know precisely what we have to use in term of platform and software.

4. Team Cohesion (TEAM): *extra high*. Each member of the team knew the others some year ago in university. So everyone have similar idea our the project.

5. Process Maturity (PMAT): *nominal*. We choose the CMM method and in this metric, we think that our project is a level-2 (3) project (repeatable, because we have a lot code pre-written and we can use to speed up our work).

These element should be combined with the following formula:

$$B = 0.91 + 0.01 \times \sum_{i=1}^{5} SF_i = 0.91 + 0.01 \times 9.35 = 1.0035$$

Now we estimate the Cost Drivers. They can assumed the following value: *Very low*, *Low*, *Nominal*, *High*, *Very high*, *Extreme high*. Everyone has a different numeric value, deepening on the current effort multiplier that we are considering. Nominal means always 1, that is no change on the total effort.

1. Required Software Reliability (RELY): *nominal* because we suppose that the system could be used from a lots a people, but if the system doesn't work for some minute it isn't a big problem.

2. Data Base Size (DATA) if we suppose a database of 10000000byte and 10000 line of code, we obtain *very high* from the table.

3. Product Complexity (CPLX): Control Operations: nominal: Simple recalling of function, No particular data structure, no recursivity. Computational Operations: very low: very simple math. Device-dependent Operations: with Java we work in VM environment, with a very high level of abstraction; Data

Management Operations: simple query to MySQL; User Interface Management Operations: high, we have a quite complex and nice GUI. Now we calculate the avg:10/5=2. The final result is *low*.

4. Required Reusability (RUSE): *nominal*. We want function generic for this project only.
5. Documentation match to life-cycle needs (DOCU): *nominal*. We have to do the right quality of document.
6. Execution Time Constraint (TIME): *Nominal*. We believe that the software have to consume less than 50/100 of the resource of the system. It may be reach in case of instantly peak of request from the users.
7. Main Storage Constraint (STOR): *Nominal*.
8. Platform Volatility (PVOL): *low*. We suppose that system shouldn't change very often.
9. Analyst Capability (ACAP): *low*. This is our first evaluated project.
10. Programmer Capability (PCAP): *nominal*. we have already tried to program with java and other languages.
11. Applications Experience (AEXP): *high*. Each one know the web-developing since 2011 (at least, someone have more experience).
12. Platform Experience (PEXP): *high*: same reason.
13. Language and Tool Experience (LTEX): *nominal*: we knew Java since the last year.
14. Personnel Continuity (PCON): *very high*. The team doesn't change.
15. Use of Software Tools (TOOL): *nominal*. We use Netbeans to manage all the system, from DB to GUI.
16. Multisite Development (SITE): *nominal*.
17. Required Development Schedule (SCED): *nominal*. The deadline are known for sure.

Now we can apply these result to obtain the personal-month quantity:

$$PM = A \times SIZE^B \times \prod_{i=1}^{17} EM_i = 2.94 \times 3.248^{1.0035} \times 0.747 \approx 7.16$$

Then we can calculate the total schedule time (in months):

$$TDEV = C \times PM^F = 3.67 \times 7.16^{0.2987} \approx 6.6$$

Where C = 3.67 and F id obtained from the formula:

$$F = D + 0.2 \times 0.01 \times \sum_{i=1}^{5} SF_i = 0.28 + 0.2 \times 0.01 \times 9.35 = 0.2987$$

And where D = 0.28. Finally we can calculate the team's size with the following:

$$NUM = \frac{PM}{TDEV} = \frac{7.16}{6.6} = 1.08$$

Theorically the project requires 1 person, but our team is composed from 3 people so 7.16/3 = 2.3 months.

4 Conclusion

We can see that the COCOMO 2 prevision are greater that the real project. The difference could due to (inevitable) statics error, but finally we can be satisfied of the obtained result. Another reason could be that we have worked very hard on Christmas holiday.