Politecnico di Milano

Software Engineering 2 Project

MeteoCal

Project Reporting Document

Authors:
Andrea Celli
Stefano Cereda

February 9, 2015

Contents

Part I
Introduction

In the following document we analyze our project using two different approaches. In the first part we apply the Function Point approach in order to check whether the result is similar to the actual size of the project. In the second part we apply COCOMO formulas and compare the results with the the "real" effort spent to develop the project.

1 Size of the project

To perform the mentioned analysis we need to know the actual size of the project. 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 4372 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 Overall project time

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. $T_{RASD} = 30h + 30h = 60h$
2. $T_{DD} = 30h + 35h = 65h$
3. $T_{DEV} = 150h + 150h = 300h$

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

$T_{TOT} = T_{RASD} + T_{DD} + T_{DEV} = 60h + 65h + 300h = 425h$

Part II
Function point

In order to determine the number of LOC (lines of code) of the project we first have to find out the number of Functions Points (FP) that have to be addressed.

1. Internal Logical Files (ILF)

   (a) The system has to store information about the user and his/her participations.
   Weight: *medium*
   The complexity is medium for the following reasons:
      - The "users" table contains several fields
      - The system has to manage security settings for the account (i.e. passwords have to be encrypted and stored securely)
      - A user calendar may contain a high number of events

   (b) The system has to store events
   Weight: *medium*
   The medium weight has been chosen because the "event" table has to contain several different fields.

   (c) The system has to store notification
   Weight: *simple*
   The simple weight has been chosen because notifications are quite simple entities (with 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 and the handling of two different kind of forecasts: daily and 3 hours.

   (b) The system has to store a list of available places and their information
   Weight: *simple*
   The system has just to download the list of places at the startup of the server.

3. External Inputs (EI)

   (a) Login and Logout
   Weight: *simple*
   These operations are very straightforward. They only involve checking the correspondence between username and password and eventually shutting down the session.

(b) Registration
Weight: *simple*
These operation consists only in inserting a new tuple in the user table while performing some basic checks on the input values.

(c) Change personal settings
Weight: *simple*
These operation consists only in updating a tuple in the user table while performing some basic checks on the input values.

(d) Event creation, modification
Weight: *complex*
These operations involves the insert/update of a tuple in the event table. Quite complex issues arise because the system has to take care of different roles of users (i.e. When the creator of an event modifies it all the participants has to be informed of that and their calendar have to be updated)

(e) Search users
Weight: *simple*
The system has just to perform some queries (addressing the user table) to retrieve the list of users corresponding to the specified search key.

(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 (EI)

(a) Show "about page"
Weight: *simple*
The system has only to display a description of MeteoCal in order to inform a new user of its functionalities.

(b) Browse personal calendar
Weight: *simple*
The system retrieves events in which the user will take part and displays them.

(c) 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. Moreover it has to periodically update them.

(b) Display notifications
Weight: *complex*
The system has to detect various types of changes in the events details and generate notifications for the interested users accordingly. Thus several tables of the database are involved.

To calculate the LOC estimate we will use the following table of weights:

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

The Unadjust Function Point (UFP) is the result of the following contributes:

- **ILF** $= 1 * simple + 2 * medium = 7 + 10 + 10 = 27$
- **EIF** $= 1 * simple + 1 * complex = 5 + 10 = 15$
- **EI** $= 4 * simple + 2 * complex = 4 * 3 + 2 * 6 = 24$
- **EIQ** $= 2 * simple + 1 * medium = 2 * 3 + 1 * 4 = 10$
- **EO** $= 2 * complex = 2 * 7 = 14$

UFP $= 27 + 15 + 24 + 0 + 14 = 90FPs$
In order to obtain the estimation of the lines of code we have to use the following simple formula:

$$LOC = AVG * UFP$$

Where AVG is a language dependent factor. We use the AVG specific for J2EE, which is the programming language mainly used to develop the project. The coefficient was taken from: `http://www.qsm.com/resources/function-point-languages-table` (we used the value in the Avg column)
The final estimation turns out to be:

$$LOC = 46 * 90 = 4140$$

Given an actual size of 4372 LOC the estimation turns out to be really precise. There is a difference of just 232 lines, which is the 5,3% of the total real size.

Part III

COCOMO II