Department of

Computer Science and Engineering

MeteoCal

Project development for the 2014 Software Engineering 2 Course

Supervisor: **Prof. Di Nitto**

Authors

Andrea Bignoli
Matr. 837493
andrea.bignoli@gmail.com

Leonardo Cella
Matr. 838074
leonardocella@gmail.com

Project

Report

Digest

The Project Reporting Document is based on an analysis made with two different metrics. The first one is the Function Points (FP), which is used to estimate the code size. The second is the COCOMO II that is used to estimate the efforts required in the development of a project by taking in account: characteristics of people, products and process. In particular we are going to compare the COCOMO calculations and the real data relative to our development of project MeteoCal.

Problem description

The X company wants to offer a new weather based online calendar for helping people scheduling their personal events avoiding bad weather conditions in case of outdoor activities. Users, once registered, should be able to create, delete and update events. An event should contain information about when and where the event will take place, whether the event will be indoor or outdoor. During event creation, any number of registered users can be invited. Only the organizer will be able to update or delete the event. Invited users can only accept or decline the invitation. Whenever an event is saved, the system should enrich the event with weather forecast information (if available). Moreover, it should notify all event participants one day before the event in case of bad weather conditions for outdoor events. Notifications are received by the users when they log into the system.

Table of Contents

1. Counting with Function Points FP

1.1. Brief Introduction

The Function Point estimation approach, is based on the amount of functionalities in a software and their complexity. Function Points estimators are useful since they are based on informations that are available early in the project life cycle.

To perform this estimation we've based our parameters on the following tables, taken from COCOMO II, Model Definition Manual at:

http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf

- This first schema is used for determining complexity-level function counts. It allow us to classify each functionality into Low, Average and High complexity levels.

Table 2. FP Counting Weights

For Internal Logical Files and External Interface Files

	Data Elements		
Record Elements	**1 - 19**	**20 - 50**	**51+**
1	Low	Low	Avg.
2 - 5	Low	Avg.	High
6+	Avg.	High	High

For External Output and External Inquiry

	Data Elements		
File Types	**1 - 5**	**6 - 19**	**20+**
0 or 1	Low	Low	Avg.
2 - 3	Low	Avg.	High
4+	Avg.	High	High

For External Input

	Data Elements		
File Types	**1 - 4**	**5 - 15**	**16+**
0 or 1	Low	Low	Avg.
2 - 3	Low	Avg.	High
3+	Avg.	High	High

- This second one, defines the weights values that we've to use to perform the FP value.

Table 3. UFP Complexity Weights

	Complexity-Weight		
Function Type	**Low**	**Average**	**High**
Internal Logical Files	7	10	15
External Interfaces Files	5	7	10
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6

- Then we need to convert the FP to lines of code. We will be using the table found at this URL: http://www.qsm.com/resources/function-point-languages-table

[SLOC/UFP] J2EE: 46

1.2. FP Estimation

1.2.1. Internal Logic Files

The application includes a number of ILFs that will be used to store the information about users, weather conditions, and events. Now we analyse this section more in detail.
The system stores few information about users, it saves: an username as a String, a password that is also a String and finally it may save an email that is not mandatory.
To provide updated weather forecasts, the system has to save the location, the effective weather forecasts already retrieved and the date/time related to an event.
And finally the system has to manage events, that are mapped to several entities: in particular for them the system has to save the creator, the list of invitations and of participants. It has also to manage a data structure which describes the relation between cities and countries , in order to allow only creation of consistent information. Moreover for events, the system has to automatically produce notification and suggested change if needed.

By resuming this description the application has to handle with two low-complex logic files and a high complex one.

ILF	*Complexity*	*FP*
User	Average	10
Event	High	15
Geographic Structure	Low	7
Total:		32

1.2.2. External Logic Files

The application has to manage the weather forecasts acquisition from an external service based on date, time and location. The data obtained using the service APIs, returned in JSON or XML format, must be then converted to be used by our system. Moreover the selected service allows two different types of forecast requests:

- Next three days: forecasts every three hours
- Next sixteen days: daily forecasts

Our system will make use of both when needed. The results will be conveniently merged and converted to a proper format.

Due to the complexity of this interaction and because of the quantity of instances involved in the process it's reasonable to classify this logic file as a complex type.

ELF	*Complexity*	*FP*
Weather Forecasts	High	10
Total:		10

1.2.3. External Inputs

The application interacts with the user to allow him/her to:

- Login/Logout: these are simple operations related only to the session manager, so their total contribute is equal to: 6 Fps.
- Create/Delete / Update an Event: Each of these operations is made on a single form but the entities involved in this kind of operations are at least six(depends on if there are or not guests) so these operations are classified as complex and their contribute is of 18 FPs.
- Participate / Cancel Participation to an Event:These operations are not complex as the previous one in fact they are related only to two entities, the event and the invitation. They are classified as low-complexity and then worth 6 FPs.
- Invite User: This operation map a relation between Users and Events as before and its contribute is of 3 Fps.
- Accept / Decline invitation:
  The first operation is equivalent to participating to an event since we intentionally made it so an invitation is still visible after being accepted, so it's contribute is of 0 Fps. A declined invitation stops being shown in the invitation management page of a user. The complexity of the last operation is low and its FP contribute is of 3.
- Search Event / User: these operations consist of searching event and user entries in our database that match given parameters. These operations are complex and give a contribute of 6 FPs.
- Insert/Update User: these operations are translated in registration and account settings editing operated by the owner. Both of them are done on a single entity and then are simple, so their contribute is of 6 Fps.

The results are resumed in the following table:

EI	*Complexity*	*FP*
Login / Logout	Low	2 x 3
Create/Delete / Update	High	3 x 6
Participate / Cancel Participation	Low	2 x 3
Invite User	Low	3
Decline Invitation	Low	3
Search Event / User	Low	2 x 3
Insert/Update User	Low	2 x 3
Total:		48

1.2.4. External Inquiries

The application allows users to request informations about their profile, the events they created or are participating to.

Moreover the calendar of another user provided according to his privacy settings. This operation is more complex since it takes into account both calendar and events visibility to produce a proper output.

Users can also request a list of their invitations and notification, and finally they can request details about specific events.

The results are resumed in the following table:

EQ	*Complexity*	*FP*
User Profile	Low	3
Events – Created and Participating To	Low	2 x 3
Other User Calendar	High	6
Invitations	Low	3
Notifications	Low	3
Event Details	Low	3
Total:		24

1.2.5. External Outputs

The application allows the creation of suggested changes. This is a complex operation because it has to perform compatibility evaluations based on the adverse conditions selected for an event and the weather forecast related to it.

EO	*Complexity*	*FP*
Suggested Schedule Changes	High	7
Total:		7

1.2.6. Resuming

By summing up all these numerical values we get a total estimation of 109 FPs. This value is used to get the estimation of the number of lines of code.
By using the parameter that was wrote in the first paragraph, we get a SLOC equal to 5014.

The following table resumes our estimations:

Function Type	*Value*
Internal Logic Files	32
External Logic Files	10
External Inputs	48
External Inquiries	24
External Outputs	7
Total:	121

1.3. Evaluation of the estimation

In this paragraph we compare the result given by the estimation with the real size of our project.

The estimated number of lines of code is:

```
LOC = 121 * 46 = 5566
```

Our project is made of 7116 lines of code. This calculus was made by considering only the effective lines of code. We have not considered comments, css and lines with only parenthesis.

In our opinion this difference is caused by other functionalities that aren't fully described by this classification, such as filters implemented to provide the best type of page navigation while appropriately managing application resources. Moreover the analysis does not capture completely the complexity of the interaction with the external service, that, in the provided implementation, can actually perform operations that exceed the initial problem requirements.

In fact without these last functionalities the effective number of lines of code is roughly equal to 6029 that is much closer to the estimated value.

2. Effort Estimation: COCOMO II

2.1. Brief Introduction

This estimation is achieved through a complex, non linear model that takes in account the characteristics of the product but also of people and process.

In this estimation, we declare that we've used also already existing components (e.g. for the graphic we used a lot of components offered by PrimeFaces).

All the tables used in this analysis have been taken from COCOMO II, Model Definition Manual at:

http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf

2.2. Scale Drivers

Table 10. Scale Factor Values, SF_j, for COCOMO II Models

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

These values are evaluated according to the following table:

- Precedentedness:

  It reflects the previous experience that we had with this kind of projects. Since for us this was the first experience using this framework and these development methodologies, this value will be low.
- Development flexibility:

  It reflects the degree of flexibility in the development process. The professors set the general specifications without going too much in detail, for this reason this value will be high.

- Risk resolution:
  Reflects the extent of risk analysis carried out. Thanks to filters, and security access mostly of the risks were eliminated then this value will be very high.
- Team cohesion:
  Reflects how well the development team know each other and work together. In our case we had some problems, in particular for the difference of working time and initial synchronization issues. Nonetheless, we overcame those difficulties by thoroughly describing guidelines and goals in our development process. Since this approach was successful the final value for this attribute is high.
- Process maturity:
  This was evaluated around the 18 Key Process Area (KPAs) in the SEI Capability Model. Because of the goals were consistently achieved these values will be set to high, level 3.

The results are resumed in the following table:

Scale Driver	Factor	*Value*
Precedentedness	Low	4.96
Development Flexibility	High	2.03
Risk Resolution	Very High	1.41
Team Cohesion	High	2.19
Process Maturity	High	3.12
Total:		13.71

2.3. Cost Drivers

- Required Software Reliability:

  This measure is not too high in particular because of software failures are translated with loss of appointments or events, so this parameter is setted to low. Software failures don't have critical consequences.

Table 17. RELY Cost Driver

RELY Descriptors:	slight inconven-ience	low, easily recoverable losses	moderate, easily recoverable losses	high financial loss	risk to human life	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.82	0.92	1.00	1.10	1.26	n/a

- Data Base Size:

  It translates the effects that large data have in our application. Our test database size is equal to 224.0 KB and the program size is equal to 7116 SLOC, the division D/P = 31.48 and then this parameter has a nominal value.

Table 18. DATA Cost Driver

DATA* Descriptors		Testing DB bytes/Pgm SLOC < 10	10 ≤ D/P < 100	100 ≤ D/P < 1000	D/P ≥ 1000	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	0.90	1.00	1.14	1.28	n/a

- Product Complexity:

  Set to high according to the new COCOMO II CPLEX rating scale.

Table 20. CPLX Cost Driver

Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.73	0.87	1.00	1.17	1.34	1.74

- Required Reusability:

  In our project there are different reusable components since our aim was to design the system as modular as possible. One of the most evident examples is the top bar that can be found at the top of each web page in the system. This parameter is therefore set to high.

Table 21. RUSE Cost Driver

RUSE Descriptors:		none	across project	across program	across product line	across multiple product lines
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	0.95	1.00	1.07	1.15	1.24

- Documentation match to life-cycle needs:
  This parameter describes the relation between the provided documentation and the application requirements. Its suitability is setted to nominal since each aspect of our system to be described has been expressed in the RASD or in the DD. On the other hand, there is no part of those document unrelated to the actual phase of the development the document is addressed to.

Table 22. DOCU Cost Driver

DOCU Descriptors:	Many life-cycle needs uncovered	Some life-cycle needs uncovered.	Right-sized to life-cycle needs	Excessive for life-cycle needs	Very excessive for life-cycle needs	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.81	0.91	1.00	1.11	1.23	n/a

- Execution Time Constraint:
  In our case this parameter is not relevant so is reasonable to set it as very low

Table 23. TIME Cost Driver

TIME Descriptors:			≤ 50% use of available execution time	70% use of available execution time	85% use of available execution time	95% use of available execution time
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	n/a	1.00	1.11	1.29	1.63

- Main Storage Constraint:
  This parameter represents the degree of main storage constraint. In our application this parameter is not relevant so is setted as very low.

Table 24. STOR Cost Driver

STOR Descriptors:			≤ 50% use of available storage	70% use of available storage	85% use of available storage	95% use of available storage
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	n/a	1.00	1.05	1.17	1.46

- Platform Volatility:
  In our application is reasonable to consider as platforms the DBMS, the operating system, the browser that perform injections and the hardware as far as the environment concerns. We have to consider also the compiler and the webServer that has taken an important role in the developing phase. The platform shouldn't change too often so this value is set to low.

Table 25. PVOL Cost Driver

PVOL Descriptors:		Major change every 12 mo.; Minor change every 1 mo.	Major: 6 mo.; Minor: 2 wk.	Major: 2 mo.;Minor: 1 wk.	Major: 2 wk.;Minor: 2 days	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	0.87	1.00	1.15	1.30	n/a

- Analyst Capability:

  Design and analysis abilities should be set to high, since we intentionally dedicated a lot of effort in analysing the problem requirements and its potential integration in a real word scenario. In particular, not only we can grant that the requirements have been correctly studied and accomplished, but also that our design makes our application actually useful for an end user, providing each of the basic functionalities he may need. In particular we resolved any ambiguity present in the initial description and explained our solution in the RASD.

Table 26. ACAP Cost Driver

ACAP Descriptors:	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.42	1.19	1.00	0.85	0.71	n/a

- Programmer Capability:

  This parameter is evaluated according to our degree of cooperation, due to some small problems on it, this value it's set to high.

Table 27. PCAP Cost Driver

PCAP Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.34	1.15	1.00	0.88	0.76	n/a

- Application Experience:

  Our project experience is evaluated according to our previous experience in web projects and also according to our abilities in programming in Java and most importantly in the Java EE framework. Since this is our first experience in this typology of project this value is equal to low.

Table 29. APEX Cost Driver

APEX Descriptors:	≤ 2 months	6 months	1 year	3 years	6 years	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.22	1.10	1.00	0.88	0.81	n/a

- Platform Experience:

  Our average knowledges about platforms as: databases, user interfaces and server-side development are around 1 year, so this parameter is setted as nominal.

Table 30. PLEX Cost Driver

PLEX Descriptors:	≤ 2 months	6 months	1 year	3 years	6 year	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.19	1.09	1.00	0.91	0.85	n/a

- Language and Tool Experience:
  This parameter reflects the same experience of the previous one, to it's setted to nominal too.

Table 31. LTEX Cost Driver

LTEX Descriptors:	≤ 2 months	6 months	1 year	3 years	6 year	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.20	1.09	1.00	0.91	0.84	

- Personnel continuity:
  This parameter is relevant in particular since in the current case our available time is less than half a year. For this reason we set it to very low.

Table 28. PCON Cost Driver

PCON Descriptors:	48% / year	24% / year	12% / year	6% / year	3% / year	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.29	1.12	1.00	0.90	0.81	

- Usage of Software Tools:
  We used NetBeans with Maven to manage dependencies of our project as libraries and development kits and Git for the repository management. The most appropriate value is nominal.

Table 32. TOOL Cost Driver

TOOL Descriptors	edit, code, debug	simple, frontend, backend CASE, little integration	basic life-cycle tools, moderately integrated	strong, mature life-cycle tools, moderately integrated	strong, mature, proactive life-cycle tools, well integrated with processes, methods, reuse	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.17	1.09	1.00	0.90	0.78	n/a

- Multisite development:
  This parameter reflects how we handled the distribution of development over distance and multiple platforms. We've used phones,mail and Skype also with screen sharing, so this value is setted to extra high

Table 33. SITE Cost Driver

SITE: Collocation Descriptors: SITE: Communications Descriptors:	Inter-national Some phone, mail	Multi-city and Multi-company Individual phone, FAX	Multi-city or Multi-company Narrow band email	Same city or metro. area Wideband electronic communicat ion.	Same building or complex Wideband elect. comm., occasional video conf.	Fully collocated Interactive multimedia
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.22	1.09	1.00	0.93	0.86	0.80

- Required development schedule:
  Our efforts were well distributed over the available development time, but regardless of this fact, the implementation required high efforts at the later phases. Mainly this is due to the fact that we expanded the initial problem description in the more complex and profitable way for a real world application. For these reason this parameter should be set to high.

Table 34. SCED Cost Driver

SCED Descriptors	75% of nominal	85% of nominal	100% of nominal	130% of nominal	160% of nominal	
Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multiplier	1.43	1.14	1.00	1.00	1.00	n/a

Our results are expressed in the following table:

Scale Driver	Factor	*Value*
Required Software Reliability	Low	0.92
Data Base Size	Nominal	1.00
Product Complexity	High	1.17
Required Reusability	High	1.07
Documentation match to life-cycle needs	Nominal	1.00
Execution Time Constraint	Very Low	n/a
Main Storage Constraint	Very Low	n/a
Platform Volatility	Low	0.87
Analyst Capability	High	0.85
Programmer Capability	High	0.88
Application Experience	Low	1.10
Platform Experience	Nominal	1.00
Language and Tool Experience	Nominal	1.00
Personnel continuity:	Very Low	1.29
Usage of Software Tools	Nominal	1.00
Multisite development	Extra High	0.80
Required development schedule	High	1.00
Product:		0.85

2.4. Effort Equation

This final equation gives us the effort estimation measured in Person-Months (PM)

$$Effort := A * EAF * KSLOC^{E}$$

Where:

A → 2.94 (for COCOMO.2000)
EAF → product of all the cost drivers, equal to : 0.85 ;
E → exponent derived from Scale Drivers. Is calculated as:
B + 0.01 * sum{i} SF[i] := B + 0.01 *13.71 = 0.91 + 0.1371 = 1.0471;
in which B is equal to: 0.91 for COCOMO.2000 .
KSLOC → estimated lines of code using the FP analysis: 5.566

With this parameters we can compute the Effort value, that is equal to:

$$Effort := 2.94 * 0.85 * 5.566^{1.0471} = 15.0808 \text{ PM}$$

instead if we consider this value starting from the real KLOC (7.116) and not the estimated one, we get: 19.504 PM .

2.5. Schedule Estimation

As far as the schedule estimation we are going to use the following formula:

$$Duration := 3.67 * Effort^{F}$$

Where:

$$F := 0.28 + 0.2 * (E - B) = 0.3074$$

Follows then:

$$Duration := 3.67 * 15.0808^{0.3074} = 8.4 \approx 8$$

The duration calculated here is not even close to the actual time we had to our disposal for this project, that could be estimated being around 3 months. We suppose that this may be due to the fact that COMODO assumptions don't apply fully to our situation in the academic context. We can absolutely state nevertheless, that the time required for the development of the project was actually really high over the limited duration of the implementation period. So, even if the estimated duration is really high compared to the actual time that we had available, we can certainly see some reasons explaining such a difference.

As for the required number of people the estimation is:

$$P = Effort / Duration = 1.79 \approx 2$$

This result, since we are actually a team of two persons, is coherent with the reality of the development environment.