Politecnico di Milano
M. Sc. Course in IT Engineering

Software Engineering 2 – Internal Project
Prof. Elisabetta di Nitto – a.y. 2013/2014

Project Reporting Document

VERSION 1

by
Edoardo Mondoni (816283)

Table of contents

Index of tables

Preamble

This is the Project Reporting Document for the Software Engineering 2 2013 internal project, named *TravelDream*. It contains a concise *a posteriori* analysis of the project, in terms of its size and of the effort it required. The two most prominent and effective methods for size and effort estimation – i.e. the Function Point approach and the COCOMO model, respectively – have been applied to the project after its termination to check for the consistency of their output.

The remainder of the document is organized as follows:

- **Section 1** sums up some relevant facts related to *TravelDream Agent* for what concerns its size and the effort it took to complete it;
- **Section 2** details the application of the Function Point approach to the project's requirements;
- **Section 3** reports on the output of the COCOMO model when used on *TravelDream Agent*'s actual size.

1 Facts

The application of the Function Points and of the COCOMO models in Sections 2 and 3 of this document has been carried out with the intention of comparing their results with its actual size and with the effort it required. In order to do so, it is necessary to provide a summary of the most important figures and stats regarding *TravelDream Agent*.

1.1 Project size

Various metrics have been defined for the size of a project, both simple and complex. Given that the purpose of this phase is to correlate the forecasts output by the models, it seems reasonable to express that measure in a way that is both immediately comparable to said predictions and easy to determine. It is therefore clear, for instance, why the project's dimensions could not have been stated in Unadjusted Function Points: true, the output of the Function Point model would have been directly relatable to this measurement, but there would have been no way to calculate the final project's Function Point count (aside from *backfiring*, a risky and inaccurate practice – partially disowned even by its own inventor – allowing to obtain the project's Function Point count out of its lines of code count).

The decision has thus been made to take the **physical magnitude** of the project as a measure of its size, in terms of lines of code, rather than a foggy quantification of the amount of work it does. As is evident from Table 1, where said metrics are reported, there can be different interpretations of "line of code" either:

- Some just count the number of physical lines in the text files composing the software project, including comment lines, empty lines and non-stating lines (for example, lines containing a `}` only): this is sometimes referred to as the **PLOC (Physical Lines of Code)** count. In the writer's opinion, this cannot be held as a trustworthy indicator of the project's size, because it heavily relies on the programmers' own formatting conventions and on the programming language's syntax itself.
- Others strip comments and empty lines away from the LOC count, thus only taking "compiler-relevant" lines into consideration: this is known as the **LLOC (Logical Lines of Code)** count, which amends many of the criticalities of the PLOC count.
- Finally, an even more refined expression of the project's size can be provided by calculating the so-called **NCSS (Non-Commenting Source Statement)** count, which seems to be a Java-specific metric. Although, as it always happens in these cases, there is no standard for its definition, it is safe to say that the NCSS count is *roughly* equal to the sum of all `;`s and `{`s in the project (further detail later in this subsection). Although this metric can be considered the most accountable most of the times, **it excludes Java Annotations from the count**; considering their relevance and the frequency with which they appear in Java EE code, using this as the main size unit of measurement does not appear as a reasonable choice.

***TravelDream Agent* size metrics**			
	Java	*JavaServer Faces*	*Total*
LLOC	3428	2492	5920
NCSS	2452		4944
%	57,9 (LLOC)	42,1	

TABLE 1: *TRAVELDREAM AGENT* SIZE METRICS

Automatic calculation of these metrics has been performed by means of two different pieces of software:

- *Metrics plugin for Eclipse*. It integrates directly into Eclipse IDE and provides live, real-time metrics ranging from the simple LLOC count to more complex indicators on any Java project. More at `http://metrics2.sourceforge.net`. It has been used to obtain a reliable count of the project's LLOC.
- *JavaNCSS*. This is a standalone command-line Java software that counts the NCSS of any project, package or class provided as an argument. The output of its execution on *TravelDream Agent* is reported in the box below. As is stated on the program's home page (`http://www.kclee.de/clemens/java/javancss/`), the following pieces of code increment the count by one:
    - Package declarations (`package <packageName>`)
    - Import declarations (`import <name>`)
    - Class declarations (`<modifier> class`)
    - Interface declarations (`<modifier> interface`)
    - Field declarations (`<type> fieldName`)
    - Method declarations (`<modifier> <returnType> <methodName>`)
    - Constructor declarations (`<modifier> <ClassName>`)
    - Constructor invocations (with `new`, `this` or `super`)
    - Statements (including expressions, method calls, `if`, `else`, `while`, `do`, `for`, `switch`, `break`, `continue`, `return`, `throw`, `synchronized`, `catch`, `finally`)
    - Labels

```
$ java javancss.Main -package ./TravelDreamAgentEJB
./TravelDreamAgentEJBClient ./TravelDreamAgentWeb
Nr.   Classes Functions      NCSS  Package
  1         6        72       457  it.polimi.traveldream.ejb.beans
  2         5        28        53  it.polimi.traveldream.ejb.client.interfaces
  3         4        12        34  it.polimi.traveldream.ejb.client.shared
  4        12       143       480  it.polimi.traveldream.ejb.dto
  5        12       137       530  it.polimi.traveldream.ejb.entities
  6        13       190       860  it.polimi.traveldream.web.beans
  7         3         8        38  it.polimi.traveldream.web.beans.helper
    --------- ---------- ----------
           55       590      2452  Total

 Packages   Classes Functions      NCSS | per
---------------------------------------------------
     7.00     55.00    590.00   2452.00 | Project
               7.86     84.29    350.29 | Package
                        10.73     44.58 | Class
                                   4.16 | Function
```

As per the presentation layer, a fair amount of work has been put into the realization of the front-end: excluding its size from the project's dimension estimates could constitute a glaring error. A third software tool called CLOC (`http://cloc.sourceforge.net`) has therefore been used to get a rough estimate of the complexity of the size of the `.xhtml` files composing the project; its output has been copied below.

```
http://cloc.sourceforge.net v 1.60  T=0.15 s (129.5 files/s, 18106.0 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
JavaServer Faces                19            165              0           2492
-------------------------------------------------------------------------------
SUM:                            19            165              0           2492
-------------------------------------------------------------------------------
```

1.2 Required effort

An effort report has been provided at the end of each phase of the project, detailing the hours spent on the achievement of the goals of that stage. Table 2 summarizes those figures; keeping in mind that the COCOMO model estimates the effort until the end of the implementation phase *excluding* the time it took to carry out the requirements elicitation step, a sub-total has been provided contemplating the design and implementation phases only.

***TravelDream Agent* effort summary**	
Requirements and specification phase	81,5 hours
Design phase	34,5 hours
Implementation phase	140 hours
Acceptance testing phase	27 hours
Total effort	**283 hours**
Of which COCOMO-comparable	**174,5 hours**

TABLE 2: *TRAVELDREAM AGENT* EFFORT SUMMARY

2 Size estimation

As was anticipated in the Preamble, the elected model for the estimation of the project's size is the Function Point approach. The general guidelines used for the application of said model were those taught in class, but the assessment of each function's complexity was marginally integrated with some of the prescriptions contained in the IFPUG v4.2.1 declination of this approach.

2.1 Data-type functions

The first step in the Function Point calculation process consisted in the identification of the data-type functions, i.e. **Internal Logical Files (ILFs)** and **External Interface Files (EIFs)**. These are defined as follows:

Internal Logical File – A user identifiable group of logically related data that resides entirely within the application boundary and is maintained through External Inputs.

External Interface File – A user identifiable group of logically related data that is used for reference purposes only. The data resides entirely outside the application boundary and is maintained by another application's External Inputs.

2.1.1 Internal Logical Files

ILFs are thus basically all kinds of data types that are stored inside, maintained and managed by the application itself. A preliminary attempt to detect *TravelDream Agent*'s ILFs, fundamentally based on the observation of the JPA Entities that have been written, returned the following list:

- User-related ILFs: User
- Basic product-related ILFs: Flight, Excursion, Hotel
- Package-related ILFs: Pre-defined package, Custom package
- Purchase-related ILFs: Purchase, Flight date assignment, Excursion date assignment, Hotel date assignment

In spite of having been implemented, the *Photo* entity was not counted as an ILF since the application makes no use of it as yet.

Further research on the approach, however, revealed that considering "sibling" entities (i.e. those that share a parent entity) as independent ILFs constitutes a mistake that can heavily distort the final results, since ILFs are the function types that contribute the most to the Function Point count. As a consequence, the decision was made to re-group each of those families into a single ILF, eventually adjusting their weight later. Please note that, although they do not formally share a common parent entity, this re-grouping was performed on the date assignment ILFs, too (they are completely separate entities from the Java point of view, but are logically and essentially the same concept in three slightly different declinations). This being said, Table 3 contains the final set of ILFs identified for the project, along with a tentative rating (multiple marks in the same row denote a doubt on the corresponding ILF's rating).

	Rating		
	Simple	**Medium**	**Complex**
User	●		
Basic product	●	●	
Package		●	●
Purchase		●	
Date assignment	●		

TABLE 3: TENTATIVE ASSIGNMENT OF RATINGS TO INTERNAL LOGICAL FILES

Several authoritative sources on the Internet described a detailed procedure to obtain a correct estimate of each ILF's rating. This procedure involves the calculation of each ILF's **Data Element Types (DETs)** and **Record Element Types (RETs)**, which were however not explained in class. Due to a lack of time, it wasn't possible to examine the subject in depth; in short, DETs are basically the number of *user-recognizable* fields inside the ILF (plus some other point in other cases, such as foreign keys), while RETs are the number of optional and mandatory subsets of the information held in the ILF.

Because none of the five detected ILFs seems to feature more than 5 RETs, and given that none of them exceeds the 19 DETs mark, it is safe to assume that all ILFs must be rated as simple, according to Table 11. This information is summarized in Table 4.

	Final rating	**Unadjusted Function Points (UFPs)**
User	Simple	7
Basic product	Simple	7
Package	Simple	7
Purchase	Simple	7
Date assignment	Simple	7
	Total	**35**

TABLE 4: FINAL RATING ASSIGNMENT AND WEIGHING FOR INTERNAL LOGICAL FILES

2.1.2 External Interface Files

EIFs are, on the other hand, all kinds of data types that the application references in one or more of its tasks, but that are located outside the software's boundary and are maintained and managed by another application.

That being the definition, it is easy to acknowledge that *TravelDream Agent* does not feature any EIFs at all. The Requirements Analysis and Specification Document states very clearly, in fact, that the application is independent and isolated from other systems, at least as of now.

2.2 Transaction-type functions

After detecting ILFs and EIFs and assigning each of them a weight, the Function Point approach requires to identify all kinds of transactions contemplated by the application, i.e. **External Inputs (EIs)**, **External Outputs (EOs)** and **External Inquiries (EQs)**. Their definitions follow:

External Input – An elementary process in which data crosses the boundary from outside to inside. The data may come from a data input screen or another application. The data may be used to maintain one or more internal logical files.

External Output – An elementary process in which derived data passes across the boundary from inside to outside. Additionally, an EO may update an ILF. The data creates reports or output files sent to other applications.

External Inquiry – An elementary process with both input and output components that result in data retrieval from one or more internal logical files and external interface files. The input process does not update or maintain any ILFs or EIFs.

2.2.1 External Inputs

External Inputs are transactions inserting information into the application. Data entered in this way usually create, modify or delete one or more ILFs. As such, the following table reports the list of EIs that have been detected for *TravelDream Agent* and a tentative rating for each one of them (as it was for ILFs, multiple marks on the same line indicate a doubt on that EI's rating).

	Rating		
	Simple	**Medium**	**Complex**
Non role-specific			
Registration	●		
Personal details modification	●		
Administrators-only			
User creation	●		
User modification	●		
User deletion	●		
Employees-only			
Basic product creation	●	●	
Basic product modification	●	●	
Basic product deletion	●		
Pre-defined package creation		●	●
Pre-defined package modification		●	●
Delete pre-defined package	●		
Customers-only			
Package customization		●	●
Custom package modification		●	●
Custom package deletion	●		
Package purchase		●	●

TABLE 5: TENTATIVE ASSIGNMENT OF RATINGS TO EXTERNAL INPUTS

Table 12 reports the parameters that must be used in order to obtain a precise evaluation of each EI's rating. The fundamental variable is the number of **Referenced File Types (RTFs)**, which counts how many different ILFs and EIFs are involved in the input operation.

All EIs listed under the *Non role-specific* and *Administrators-only* sections have therefore correctly been rated as simple, because they only involve the creation, modification or deletion of just one ILF – of type User – and certainly do not count more than 15 DETs. Similarly, all operations on basic products performed by employees shall be considered simple, since they only reference one kind of ILF at a time and feature less than 16 DETs (please note that Basic product ILFs having three subtypes does not count as 3 RTFs, since the *Create basic product* input still handles one of those subtypes at a time).

Employee-related EIs dealing with travel packages handle 2 RTFs at a time, instead, i.e. the Travel package ILF itself and the Basic product ILFs representing the package's components. Because each package features a name, a price, a description and – of course – the list of its components, the number of DETs would be equal to 4, thus locating said EIs on the verge between the simple and medium ratings (remember that fields such as `available` or `packageID` lack the qualification of user-recognizability and must therefore not be counted as DETs). An additional DET, however, should be contemplated in light of the system's ability to notify the user of errors that occurred or of the success of the operation. This brings the total DET count to 5 for employee travel package-related EIs, thus making them medium-level.

As per all EIs involved with custom packages, the RTF count goes up to 3 (the `author` field references a User ILF, in addition to those already mentioned above). Because a custom package is in no way different from a pre-defined one, and given that the application is equally able to inform the user on the status of the procedure, these EIs feature 5 or more DETs and therefore rank as high-level. Performing a purchase requires referencing all 5 types of ILF and surely involves at least 5 DETs (date assignments, billing information, plus the system's self-awareness on the status of the process): the *Package purchase* EI should as well be counted as a high-level one.

Table 6 sums all those considerations up and reports the total number of UFPs ascribable to EIs.

	Final rating	**Unadjusted Function Points (UFPs)**
Registration	Simple	3
Personal details modification	Simple	3
User creation	Simple	3
User modification	Simple	3
User deletion	Simple	3
Basic product creation	Simple	3
Basic product modification	Simple	3
Basic product deletion	Simple	3
Pre-defined package creation	Medium	4
Pre-defined package modification	Medium	4
Pre-defined package deletion	Medium	4
Package customization	Complex	6
Custom package modification	Complex	6
Custom package deletion	Complex	6
Package purchase	Complex	6
	Total	**60**

TABLE 6: FINAL RATING ASSIGNMENT AND WEIGHING FOR EXTERNAL INPUTS

2.2.2 *External Outputs*

A transaction is an EO if it sends data outside the system boundary **and at least one** of the following holds:

- it contains at least one mathematic calculation or a formula;
- it creates derived data;
- it maintains at least one ILF;
- it alters the system's behaviour.

Because no output transaction in *TravelDream Agent* meets those requirements, no EOs are present in the application.

2.2.3 *External Inquiries*

EQs are request-response-based transactions that involve both input and output operations; they send data outside the system boundary **and** retrieve data from at least one ILF or EIF **and none** of the following hold:

- it contains at least one mathematic calculation or a formula;
- it creates derived data;
- it maintains at least one ILF;
- it alters the system's behaviour.

The application of these simple boolean rules made it easier to build a preliminary list of EQs, associating each of them with a provisional rating.

	Rating		
	Simple	**Medium**	**Complex**
Non role-specific			
Login	●		
Logout	●		
Administrators-only			
User list request	●		
Employee- and customer-reserved			
Basic product list request	●	●	
Pre-defined package list request		●	
Customers-only			
Package details request		●	
Component details request	●		
Custom packages list request		●	●
Purchases list request	●	●	

TABLE 7: TENTATIVE ASSIGNMENT OF RATINGS TO EXTERNAL INQUIRIES

Please note that the *Login* and *Logout* transactions have been listed drawing from IFPUG iTip #03 (`http://www.ifpug.org/iTips/iTip%2003Logon.pdf`), which illustrates how logon operations can act as EQs or EOs depending on the consequences they have on the system. Because no decision is taken based on the user's entered credentials (the role needs to be manually specified in the home page), the first example seems to be the most appropriate for our case. At the same time, the logout transaction cannot trigger input-based decisions because it simply requires no input on the user's part.

As it happens for EIs, rating the detected EQs requires identifying the number of different file types they reference. *Login*, *Logout* and *User list request* only involve one RTF; they surely deal with less than 20 DETs, so they can be confirmed as simple EQs. *Basic product list request* and *Component details request* also deal with one RTF only and are thus simple. *Pre-defined package list request*, *Package details request*, and *Custom packages list request* handle 2 or 3 RTFs and therefore surely rank as medium-level EQs, given that the DET count is surely smaller than 20. *Purchases list request* finally deals with all 5 ILF types and thus classifies as medium-level, since each purchase features a package, a buyer, a date assignment list and nothing more (which sets the DET count to 3).

These ratings are summarized in Table 8, along with their resulting weights:

	Final rating	Unadjusted Function Points (UFPs)
Login	Simple	3
Logout	Simple	3
User list request	Simple	3
Basic product list request	Simple	3
Pre-defined package list request	Medium	4
Package details request	Medium	4
Component details request	Simple	3
Custom packages list request	Medium	4
Purchases list request	Medium	4
	Total	**31**

TABLE 8: FINAL RATING ASSIGNMENT AND WEIGHING FOR EXTERNAL INQUIRIES

2.3 Size calculation and comparison

2.3.1 UFP count

Now that the model has been successfully applied to each of the five types of functions, the last thing to do is to calculate the total amount of Unadjusted Function Points by summing the sub-totals obtained so far, as is shown in the formula below (where w_i^X represents the weight of the i-th function of type *X*):

$$UFP = \sum_i w_i^{ILF} + \sum_j w_j^{EIF} + \sum_k w_k^{EI} + \sum_m w_m^{EO} + \sum_n w_n^{EQ}$$

So we get:

$$UFP = 35^{ILF} + 0^{EIF} + 60^{EI} + 0^{EO} + 31^{EQ} = 126$$

TravelDream Agent's size has therefore been estimated in 126 Unadjusted Function Points.

2.3.2 LOC metrics conversion

But how to compare the UFP count above with the size metrics of the actual project? Luckily enough, some companies keep a database of software projects that they use to derive conversion constants from UFP to lines of code, for a multitude of programming languages.

The most exhaustive and up-to-date collection of such data has been found at `http://www.qsm.com/resources/function-point-languages-table`. The UFP-to-LOC conversion factors are called *gearing factors* and, as is explained in the same page, they return the number of **logical** lines of code (LLOC). Table 9 reports the gearing factors for Java EE.

	Gearing factors [LLOC/UFP]			
	Average	**Median**	**Minimum**	**Maximum**
Java EE	46	49	15	67

TABLE 9: GEARING FACTORS FOR JAVA EE

The calculations return the results in Table 10, which also includes a comparison with the project's actual size (in terms of model correctness η_{UFP}) and a computation of the error $e_{\%}$ made by the model, according to the following formulae:

$$\eta_{UFP} = \frac{LLOC_{estimate}}{LLOC_{actual}}$$

$$e_{\%} = 100 \cdot (\eta_{UFP} - 1)$$

	Minimum	**Average**	**Maximum**	
$LLOC_{estimate}$	1890	5796	8442	Remember: from Subsection 1.1 $LLOC_{actual} = 5920$
η_{UFP}	0,32	0,98	1,43	
$e_{\%}$	-68%	-2%	+43%	

TABLE 10: FINAL ESTIMATIONS AND ACTUAL SIZE COMPARISONS

As is evident from the results above, the model – together with the gearing factors – estimated the number of LLOC with an error of about -2% (obviously considering the average gearing factor), which can be considered a very good result. It could not be verified whether the Java EE gearing factor includes the front-end-related lines of code or is intended to return the Java LLOC count only, in which case the error would have been considerably higher (+69%); given the precision of the estimate, though, the former hypothesis seems most likely.

2.4 Reference tables

This subsection reports the reference tables that have been used to rate and subsequently weigh the functions identified during the application of the Function Point method.

2.4.1 Rating tables

Rating tables associate return a function's rating depending on its number of RETs (in case of ILFs and EIFs) or RTFs (in case of EIs, EOs, and EQs) and DETs.

ILFs and EIFs	*Data Element Types (DETs)*		
Record Element Types (RETs)	**1 to 19**	**20 to 50**	**More than 50**
1	Simple	Simple	Medium
2 to 5	Simple	Medium	Complex
More than 5	Medium	Complex	Complex

TABLE 11: ILFS AND EIFS RATING REFERENCE TABLE

EIs	*Data Element Types (DETs)*		
Referenced File Types (RFTs)	**1 to 4**	**5 to 15**	**More than 15**
1	Simple	Simple	Medium
2	Simple	Medium	Complex
More than 2	Medium	Complex	Complex

TABLE 12: EIS RATING REFERENCE TABLE

EOs and EQs	*Data Element Types (DETs)*		
Referenced File Types (RFTs)	**1 to 5**	**6 to 19**	**More than 19**
1	Simple	Simple	Medium
2 or 3	Simple	Medium	Complex
More than 3	Medium	Complex	Complex

TABLE 13: EOS AND EQS RATING REFERENCE TABLE

2.4.2 Weighing table

Once a function has been rated according to the previous tables, its weight in terms of Unadjusted Function Points can be obtained from the table below.

	Rating [Unadjusted Function Points (UFPs)]		
Function type	**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

TABLE 14: FUNCTION WEIGHING REFERENCE TABLE

3 Effort estimation

The Constructive Cost Model (COCOMO) for the estimation of the effort required by software projects has been developed in the late Seventies and ultimately published in 1981. Several improvements have been made during the years, and a revised version of this approach, named COCOMO II, was published in 2000. *TravelDream Agent*'s effort estimation has been carried out with both versions of this model, in order to evaluate the different results they return.

3.1 Original COCOMO

The original version of COCOMO came in three different and increasingly complex declinations:

- **Basic COCOMO** relied solely on the project's size to determine the required effort, according to pre-defined equations and coefficients; its projections proved to be off by less than 20% in 25% of the cases.
- **Intermediate COCOMO** corrected the results obtained in the Basic version by altering some of the coefficients and introducing an array of 15 factors (called *cost drivers*), focused both on the product's and on the development process's features, each of which had to be rated on a scale from Very low to Extra high and then translated into a numeric value according to pre-defined tables; its assessments proved to be off by less than 20% in 68% of the cases.
- **Detailed COCOMO** went even further to evaluate each cost driver's impact on the different phases of the development process; its improvements were, however, marginal, since only 70% of the estimations it provided were off by less than 20%.

However, those three versions share a lot of aspects:

- *Project size.* The project size is always expressed in **thousands of Delivered Source Instructions (kDSI)**, which is another name for LLOC.
- *Development phases.* The estimation refers solely to the effort required by the Design, Implementation and Testing (+ Integration) phases.
- *Unit of measurement.* The fundamental equation provided by COCOMO always returns the effort in terms of person-months, which can eventually be translated into calendar months by means of another formula.
- *Project classification.* The project needs to be informally classified as *organic*, *embedded* or *semi-detached*, mainly according to the team's size, the product's precedentedness and its degree of independence from other software products. Coefficients vary based on this classification.

Given the limited additional precision introduced by Detailed COCOMO over the Intermediate version's output, the decision was made to ignore this last approach and to stick to its two simpler declinations.

3.1.1 *Preliminary considerations on the project*

TravelDream Agent can be considered an organic project. The development team is, in fact, as small as it can get (being composed of one person only) and the product is not expected to interact with other systems.

COCOMO requires the LLOC count as an input, but what lines of code should actually be counted: Java-only or Java and JSF combined? The latter seems the most reasonable (in the end, *it took time and effort* to write the front-end), but the estimations will be performed on both values.

3.1.2 *Basic COCOMO*

According to the equations and coefficients laid out in Paragraph 3.3.1, Basic COCOMO returns the following estimations for *TravelDream Agent*; Table 15 also indicates the model's correctness η_b (the closer to 1, the better the model) and its error $e_{\%}$ (the closer to 0, the better the model), which were calculated based on the following formulae:

$$\eta_b = \frac{E_{estimate}}{E_{actual}}$$

$$e_{\%} = 100 \cdot (\eta_b - 1)$$

The conversion from person-months to person-hours has been performed keeping in mind that both Original COCOMO and COCOMO II hold 1 person-month to correspond to 152 person-hours.

	Project size	
	Java code only	**Java + JSF**
E [person-months]	8,75	15,53
T [months]	5,7	7,1
E [person-hours]	1330	2361
η_b	7,62	13,53
$e_{\%}$	+662%	+1253%

TABLE 15: BASIC COCOMO ESTIMATIONS

It is evident that the model is off by several times: the estimate is not even remotely accurate.

3.1.3 *Intermediate COCOMO*

Given the tremendous error of the Basic COCOMO approach, an attempt to correct it with the Intermediate version has been made. As was anticipated, Intermediate COCOMO requires to rate 15 cost drivers, thus obtaining 15 different coefficients: their product is called **Effort Adjustment Factor (EAF)** and is used to calculate the effort estimation.

Table 16 reports the ratings assigned to each one of the cost drivers and the resulting EAF.

Cost driver	**Rating**	**Coefficient**
Required software reliability	Very low	0,75
Size of the database	Low	0,94
Complexity of the product	Very low	0,7
Run-time performance constraints	Nominal	1

Cost driver	**Rating**	**Coefficient**
Memory constraints	Nominal	1
Volatility of the VM environment	Low	0,87
Required turnaround time	Nominal	1
Analyst capability	Low	1,19
Software engineer capability	Very low	1,29
Applications experience	Low	1,13
VM experience	Very low	1,21
Programming language experience	Nominal	1
Use of software tools	High	0,91
Application of SW engineering methods	High	0,91
Required development schedule	High	1,04
	Effort Adjustment Factor (EAF)	**0,602**

TABLE 16: COST DRIVERS ASSESSMENT

As a consequence, the model outputs the following results (η_b and $e_\%$ calculated as described in Paragraph 3.1.2).

	Project size	
	Java code only	**Java + JSF**
E *[person-months]*	7,02	13,23
T *[months]*	5,24	6,17
E *[person-hours]*	1067	2011
η_b	6,11	11,52
$e_\%$	+511%	+1052%

TABLE 17: INTERMEDIATE COCOMO ESTIMATIONS

The results are practically unchanged: the model is still off by several times the actual effort spent.

3.2 COCOMO II

COCOMO II eliminates the distinction among Basic, Intermediate and Detailed. To be honest, a Post-Architecture and an Early Design declination have been developed, but the latter is intended for early-stage estimations and is therefore not of interest. The project categorization into organic, semi-detached and embedded is also gone, since its effects have been moved to a series of more flexible coefficients used in the calculation of effort and calendar time.

The first step requires to evaluate the rating for each one of the Scale Factors concurring to determine the project size's exponent. The results of this activity are summarized in Table 18.

Scale factor	**Rating**	**Score**
Precedentedness (PREC)	Nominal	3,72
Development flexibility (FLEX)	Very high	1,01
Architecture/Risk resolution (RESL)	Nominal	4,24

Scale factor	**Rating**	**Score**
Team cohesion (TEAM)	Extra high	0
Process maturity (PMAT)	Low	6,24
$\sum_{j=1}^{5} SF_j$		**15,21**

TABLE 18: COCOMO II SCALE FACTORS RATING AND CALCULATION

The next step consists in the assessment of the 16 Effort Multipliers and in the determination of the product of their relative coefficients, which is reported in Table 19.

Effort Multiplier	**Rating**	**Coefficient**
Required software reliability (RELY)	Very low	0,82
Database size (DATA)	Low	0,90
Product complexity (CPLX)	Very low	0,73
Reusability (RUSE)	Low	0,95
Documentation suitability (DOCU)	Nominal	1
Execution time constraint (TIME)	Nominal	1
Main storage constraint (STOR)	Nominal	1
Platform volatility (PVOL)	Low	0,87
Analyst capability (ACAP)	Low	1,19
Programmer capability (PCAP)	Nominal	1
Personnel continuity (PCON)	Very high	0,81
Applications experience (APEX)	Nominal	1
Platform experience (PLEX)	Very low	1,19
Language and tool experience (LTEX)	Nominal	1
Use of software tools (TOOL)	Very low	1,17
Multisite development (SITE)	Extra high	0,80
Required development schedule (SCED)	Nominal	1
$\prod_{i=1}^{16} EM_i$		**0,478**

TABLE 19: COCOMO II EFFORT MULTIPLIERS RATING AND CALCULATION

All variables having been calculated, the COCOMO II equations (Paragraph 3.3.3) can be applied.

	Project size	
	Java code only	**Java + JSF**
E *[person-months]*	5,2	9,29
T *[months]*	5,26	6,17
E *[person-hours]*	790	1412
η_b	4,53	8,09
$e_{\%}$	+353%	+709%

TABLE 20: COCOMO II ESTIMATIONS

We can therefore conclude that, despite the COCOMO II model being indeed the most precise of the three effort estimation approaches used in this document, its accuracy is strongly questionable. It should be noted, however, that the project base upon which the model was developed did not include any academic project, and that single-person teams are not contemplated (one of the six basic hypotheses of the Original COCOMO approach was that the project be developed by a "small and expert team").

3.3 Model reference

This subsection reports the reference formulae and tables of which the model consists.

3.3.1 *Original COCOMO, Basic version*

$$E = a_b \cdot S^{b_b} \qquad T = c_b \cdot E^{d_b}$$

(*S* being the project's size in kLLOC)

Basic	*Coefficient*			
Project type	a_b	b_b	c_b	d_b
Organic	2,4	1,05	2,5	0,38
Semi-detached	3,0	1,12	2,5	0,35
Embedded	3,6	1,20	2,5	0,32

TABLE 21: ORIGINAL COCOMO COEFFICIENTS, BASIC VERSION

3.3.2 *Original COCOMO, Intermediate version*

$$E = a_i \cdot S^{b_i} \cdot EAF \qquad EAF = \prod_{j=1}^{15} CD_j \qquad T = c_i \cdot E^{d_i}$$

(*S* being the project's size in kLLOC, CD_j being the coefficient of the *j*-th cost driver)

Intermediate	*Coefficient*			
Project type	a_i	b_i	c_i	d_i
Organic	3,2	1,05	As in Basic COCOMO	
Semi-detached	3,0	1,12		
Embedded	2,8	1,20		

TABLE 22: ORIGINAL COCOMO COEFFICIENTS, INTERMEDIATE VERSION

Intermediate	*Rating* (*VL: very low, L: low, N: nominal, H: high, VH: very high, EH: extra high*)					
Cost driver	**VL**	**L**	**N**	**H**	**VH**	**EH**
Product attributes						
Required software reliability	0,75	0,88	1	1,15	1,40	
Size of the database		0,94	1	1,08	1,16	
Complexity of the product	0,70	0,85	1	1,15	1,30	1,65
Hardware attributes						
Run-time performance constraints			1	1,11	1,30	1,66
Memory constraints			1	1,06	1,21	1,56

Volatility of the VM environment		0,87	1	1,15	1,30	
Required turnaround time		0,87	1	1,07	1,15	
Personnel attributes						
Analyst capability	1,46	1,19	1	0,86	0,71	
Software engineer capability	1,42	1,17	1	0,86	0,70	
Applications experience	1,29	1,13	1	0,91	0,82	
VM experience	1,21	1,10	1	0,90		
Programming language experience	1,14	1,07	1	0,95		
Project attributes						
Use of software tools	1,24	1,10	1	0,91	0,83	
Application of SW engineering methods	1,24	1,10	1	0,91	0,82	
Required development schedule	1,23	1,08	1	1,04	1,10	

TABLE 23: ORIGINAL COCOMO COST DRIVERS, INTERMEDIATE VERSION

3.3.3 COCOMO II Post-Architecture model

$$E = 2,94 \cdot S^{e} \cdot \prod_{i=1}^{16} EM_{i} \qquad \text{where } e = 0,91 + 0,01 \cdot \sum_{j=1}^{5} SF_{j}$$

$$T = 3,67 \cdot E^{f} \qquad \text{where } f = 0,28 + 0,2 \cdot (e - 0,91)$$

(S being the project's size in kLLOC, EM_i being the i-th Effort Multiplier, SF_j being the j-th Scale Factor)

	Rating *(VL: very low, L: low, N: nominal, H: high, VH: very high, EH: extra high)*					
Scale factor	**VL**	**L**	**N**	**H**	**VH**	**EH**
Precedentedness (PREC)	6,20	4,96	3,72	2,48	1,24	0
Development flexibility (FLEX)	5,07	4,05	3,04	2,03	1,01	0
Architecture/Risk resolution (RESL)	7,07	5,65	4,24	2,83	1,41	0
Team cohesion (TEAM)	5,48	4,38	3,29	2,19	1,10	0
Process maturity (PMAT)	7,80	6,24	4,68	3,12	1,56	0

TABLE 24: COCOMO II SCALE FACTORS

	Rating *(VL: very low, L: low, N: nominal, H: high, VH: very high, EH: extra high)*					
Effort Multiplier	**VL**	**L**	**N**	**H**	**VH**	**EH**
Product factors						
Required software reliability (RELY)	0,82	0,92	1	1,10	1,26	
Database size (DATA)		0,90	1	1,14	1,28	
Product complexity (CPLX)	0,73	0,87	1	1,17	1,34	1,74
Reusability (RUSE)		0,95	1	1,07	1,15	1,24
Documentation suitability (DOCU)	0,81	0,91	1	1,11	1,23	
Platform factors						
Execution time constraint (TIME)			1	1,11	1,29	1,63
Main storage constraint (STOR)			1	1,05	1,17	1,46

Platform volatility (PVOL)		0,87	1	1,15	1,30	
Personnel factors						
Analyst capability (ACAP)	1,42	1,19	1	0,85	0,71	
Programmer capability (PCAP)	1,34	1,15	1	0,88	0,76	
Personnel continuity (PCON)	1,29	1,12	1	0,90	0,81	
Applications experience (APEX)	1,22	1,10	1	0,88	0,81	
Platform experience (PLEX)	1,19	1,09	1	0,91	0,85	
Language and tool experience (LTEX)	1,20	1,09	1	0,91	0,84	
Project factors						
Use of software tools (TOOL)	1,17	1,09	1	0,90	0,78	
Multisite development (SITE)	1,22	1,09	1	0,93	0,86	0,80
Required development schedule (SCED)	1,43	1,14	1	1	1	

TABLE 25: COCOMO II EFFORT MULTIPLIERS

Miscellanea

Changelog

Version 1 (February 13th, 2014)

First release of the Project Reporting Document.

Effort

A total of *23 hours* were spent on the production of this document. In particular:

- *7 hours* were reserved to an overview of what had been done in class and in its integration with other material found on the Internet;
- *6 hours* were spent on the application of the Function Point and COCOMO approaches to the project;
- *10 hours* were what it took to write this document.