TravelDream Project

Project Report

Riccardo B. Desantis - matr. 765106

Contents

Chapter 1

Introduction

1.1 Purpose

The purpose of this document is to present the results of the computation of the FP and COCOMO analysis on the TravelDream project developed for the Software Engineering 2 course.

1.2 Definition, acronyms, and abbreviations

The following acronyms will be used through the whole document:

- UFP: Unadjusted Function Points
- FP: Function Points
- ILF: Internal Logical File
- EIF: External Interface File
- RET: Record Element Type
- FTR: File Type Referenced
- DET: Data Element Type
- COCOMO: Constructive Cost Model.

1.3 References

- Progetto AA 2013-2014 (TravelDream).pdf
- rasd.pdf
- dd.pdf.

1.4 Overview

This document specifies the computation of some of the software metrics, FP and COCOMO, on the TravelDream project developed.

The document is organized in the following sections:

1. **Introduction**: this section describes the purpose of the document including its scope, glossary and related documents used to do it.

2. **Software Metrics**: contains a detailed explanation of the process used to compute the metrics, and the results of such computations.

3. **Conclusions**: contains the comparison between the actual data with the computed one.

Chapter 2

Software Metrics

2.1 FP

2.1.1 Introduction

To compute the **Function Points** (**FP**) of our project, we first consider the **Unadjusted Function Points** (**UFP**) and we then correct such computation. To compute the latter, we identify and count the **function types** and we weight them using this table:

Function Types	**Weight**		
	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

where the class of weights is chosen considering the DETs, RETs and FTRs:

- for ILF and EIF:

RETs	**DETs**		
	1-19	20-50	≥ 51
1	Simple	Simple	Medium
2-5	Simple	Medium	Complex
≥ 6	Medium	Complex	Complex

- for the External Inputs:

FTRs	**DETs**		
	1-4	5-15	≥ 16
0-1	Simple	Simple	Medium
2	Simple	Medium	Complex
≥ 3	Medium	Complex	Complex

- for the External Outputs and Inquiries:

FTRs	**DETs**		
	1-5	6-19	≥ 20
0-1	Simple	Simple	Medium
2-3	Simple	Medium	Complex
≥ 4	Medium	Complex	Complex

Given the computed UFP, we compute the FP using this formula:

$$FP = UFP \times \left(0.65 + 0.01 \times \sum_{i=1}^{14} F_i\right)$$

where the F_is are values from 0 to 5 (for increasing importance from no influence to essential) answering these questions:

1. How many data communication facilities are there?
2. How are distributed data and processing functions handled?
3. Was response time or throughput required by the user?
4. How heavily used is the current hardware platform?
5. How frequently are transactions executed?
6. What percentage of the information is entered online?
7. Was the application designed for end-user efficiency?
8. How many internal logical files are updated by on-line transaction?
9. Does the application have extensive logical or math processing?
10. Was the application developed to meet one or many user needs?
11. How difficult is conversion and installation?
12. How effective/automated are startup, backup, and recovery?
13. Was the application designed for multiple sites/organizations?
14. Was the application designed to facilitate change?

Given now the conversion table from function points to **Lines Of Code** (**LOC**) per languages, on http://www.qsm.com/resources/function-point-languages-table, we have, considering only the "high" value to stay safe:

Language	LOC per FP
C	333
C++	80
Java	134
J2EE	67
HTML	48
SQL	37
JavaScript	63

2.1.2 Computation

Considering now the project, we have these function types:

- Internal Logic File (ILF): the application stores the information about:
  - *users* (7 DETs, 2 RETs: Simple),
  - *components* (6 DETs, 4 RETs: Simple),
  - *packages* (3 DETs, 1 RET: Simple),
  - *customized packages* (5 DETs, 1 RET: Simple),

  thus we can consider 4 Simple ILFs;
- External Interface File (EIF): the application doesn't use any external data, thus there are no EIFs;
- External Inputs: the application interacts with two actor, and for each of these expose different services:
  - *login*, *logout* and *registration* (1 FTR, 2 DETs: Simple) for both customers and employees;
  - *buying of a package* (2 FTRs, at least 5 but less than 16 DETs: Medium) for a customer;
  - *customization of a package* (3 FTRs, at least 5 DETs: Complex) for a customer;
  - *insertion* and *deletion of a component* (1 FTR, less than 16 DETs: Simple) for an employee;
  - *insertion* and *deletion of a package* (2 FTR, less than 16 DETs but at least 5: Medium) for an employee.

  Thus we have 5 Simple EI, 3 Medium EI, 1 Complex EI;
- External Outputs: none, because no EIF is used;
- External Inquiry: we have
  - showing of the *profiles* of the users (1 FTR, less than 20 DETs: Simple);
  - *listing of the packages* (2 FTRs: Medium);
  - *listing of the components* (1 FTR: Simple);
  - *listing of the customized packages* (3 FTRs: Medium).

We have then 2 Simple EQ, 2 Medium EQ.

5 Simple EI, 3 Medium EI, 1 Complex EI; Summing up, considering the weights, we compute the UFP:

$$UFP = (4 \times 7) + (5 \times 3 + 3 \times 4 + 1 \times 6) + (2 \times 3 + 2 \times 4) = 75.$$

To compute the Value Adjustment Equation we consider a score of 27 (given by assigning $F_6 = 5$, $F_7 = 5$, $F_8 = 5$, $F_{10} = 4$, $F_{14} = 5$, $F_3 = 3$), then:

$$FP = 75 \times (0.65 + 0.01 \times 27) = 69$$

that corresponds to

$$LOC = 67 \times 69 = 4624 \Longrightarrow KLOC = 4.624$$

in J2EE given the table.

2.2 COCOMO

2.2.1 Introduction

COCOMO identifies three classes of applications:

- **Simple** (**Organic mode**)
- **Intermediate** (**Semi-detached mode**)
- **Complex** (**Embedded mode**)

and computes an estimation of **effort** (as in **person months**: **M**) and **calendar time** (in **months**: **T**) and the resulting number of people needed (**N**, resulting from $\frac{M}{T}$).

The estimation is computed with a hierarchy of models of increasing complexity:

- **Basic** is a static single-valued model that computes software development effort (and cost) as a function of program size expressed in estimated lines of code;
- **Intermediate** computes software development effort as function of program size and a set of "cost drivers" that include subjective assessment of product, hardware, personnel, and project attributes;
- **Advanced** incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step (analysis, design, etc.) of the software engineering process.

Considering only the basic COCOMO, given the program size **S**:

$$M = a_b S^{b_b}$$

$$T = c_b S^{d_b}$$

where

Type of application	a_b	b_b	c_b	d_b
Organic mode	2.4	1.05	2.5	0.38
Semi-detached mode	3.0	1.12	2.5	0.35
Embedded mode	3.6	1.20	2.5	0.32

2.2.2 Computation

Considering now our project as a complex application (that is considering the embedded mode then), we'll compute, using $S = KLOC$ at the previous point:

$$M = 3.6 \times 4.624^{1.2} = 22.611 \text{ person/months}$$

$$T = 2.5 \times 4.624^{0.32} = 4.08 \text{ months}$$

giving the number of requested people:

$$N = \frac{M}{T} = \frac{22.611}{4.08} = 5.542 \simeq 6 \text{ people.}$$

Chapter 3

Conclusions

3.1 Comparisons

The actual KLOC written for the project (only considering the Java files) are 4.517, instead of the computed 4.624. That is pretty close!

For the COCOMO method, we had this table of hours used:

	Riccardo B. Desantis
RASD	15 h
DD	32 h
Implementation	120 h
Testing	20 h

giving us a total of 187 h for 1 person. Considering a working environment with 8 hours per day for 5 days a week and 4 weeks a month (that is 20 days a month), it is:

$$\frac{187\ h}{8\ \frac{h}{\text{day}}} * \frac{1}{20\ \frac{\text{days}}{\text{month}}} = 1.17 \text{ months}$$

that is much less than the computed one.