Ain Shams University
Faculty of Engineering
Computer & Systems Engineering Department

Sizing and Estimation of Maintenance Software Projects

Thesis

Submitted in partial fulfillment of the Requirements for the M.Sc. Degree in Electrical Engineering
(Computer and Systems Engineering)

Submitted by

Nermeen Fawzy Ahmed Mohamed
B.Sc of Electrical Engineering
Computer and Systems Engineering Department
Faculty of Engineering, Ain Shams University, 2002

Supervised by

Prof. Dr. Abdel-Monein Abdel-Zaher Wahdan

Prof. Dr. Ashraf Mohamed Salem

Cairo 2009

APPROVAL SHEET

Acknowledgements

All praise be to ALLAH the High and All Mighty for everything I realize and what I don’t realize. I would like to express my deepest appreciation to my thesis supervisors, Professor Abdel-Moneim Abdel-Zaher Wahdan, and Professor Ashraf Mohamed Salem. I greatly valued their support, care and guidance for me through all the work phases. Thanks also for my parents for supporting me all the way till I have become engineer, and for their help and continuous motivation. Thanks to my Kid, Omar, for his pure inspiring smile. Finally thanks to my husband for providing me with unconditional support and care. I’ll never be thankful enough for him.

ABSTRACT

The objective of this thesis is to study of the existing software cost estimation techniques, and to reveal some of the drawbacks of these techniques, and to propose a new model to get rid of these drawbacks. The proposed estimation model is a modified version of the famous COCOMO model, in which we adopted the use of fuzzy logic to override some of its deficiencies. The introduction of fuzzy logic tolerates the existence of imprecision and uncertainty in measurements throughout the whole estimation process.

As the proposed cost estimation model tolerates the imprecision in the value of the cost factors, then the model is less sensitive to slight changes between projects, and makes it give better estimation results. On the other hand the original version of the intermediate COCOMO'81 does not tolerate the imprecision of the values of the cost factors.

We validated our proposed model on the COCOMO benchmark which is composed of 4 databases of software projects. Each database contains 63 different projects. The results obtained from this validation show the advantage of our proposed model over the original COCOMO model.

Keywords:

Software cost estimation, Software engineering, Fuzzy Logic.

Contents

Abbreviations

ACAP:	Analyst CAPability
AEXP:	Application EXPerience
AKDSI:	Adjusted Kilo Delivered Source Instructions
ASE:	Average Squared Error
CASE:	Computer Aided Software Engineering
CBR:	Case Based Reasoning
COCOMO:	COnstructive COst MOdel
COD:	Coefficient Of Determination
CPLX:	Product ComPLeXity
CTP:	Computational Theory of Perceptions
CW:	Computation with Words
DSI:	Delivered Source Instructions
FP:	Function Point
GSC:	General System Characteristics
GP:	Genetic Programming
IDS:	Initial Data Set
INQ:	INQuiry
ISBSG:	International Software Benchmarking Standards Group
KDSI:	Kilo Delivered Source Instructions
KLOC:	Kilo Line Of Code
LEXP:	Programming Language EXPerience
MIQ:	Machine Intelligent Quotient
MM:	Man-Month
MMRE:	Mean Magnitude Relative Error
MODP:	MODern Programming Practices
MP:	Multilayer Perceptron

MRE:	Magnitude Relative Error
MSE:	Mean Square Error
MU:	Memory Utilization
PCAP:	Programmer CAPability
RELY:	Required Software RELiabilitY
SCED:	Required Development SChEDule
SDMRE:	Standars Deviation of Magnitude Relative Error
STOR:	Main STORage Constraint
TDS:	Terminal Data Set
TE:	Total Effort of all the phases of a project
TURN:	Computer TURNaround Time
VEXP:	Virtual Machine EXPerience
VIRT:	VIRTual Machine Instability

Chapter 1
Introduction

The good management of the software development process is based on a good estimation of its cost. Without a good estimate, managers and other persons responsible for a software project will not be able to assign the budgets and the competences necessary for the project development. Thus, managers depending on the results of estimation could make serious errors by assigning inadequate time plans or insufficient budgets. These errors could be also reflected on the users. Extending the development time of a project would involve a delay in the delivery and installation of the software. The users would thus fall into economical and organizational problems, especially if they have already made up their planning and set up new structures by hiring new employees and arranging buildings for example.

Although precise estimates are difficult to establish, it is well known that software development costs represent a significant part of the project. In many software projects the development cost exceeds 80% of the total cost of the project [BOE81]. The growth of software costs as well as the difficulties that we meet to estimate them and control them still constitutes a major concern for project managers and researchers in this domain. Many problems are faced such as the increasing demands of users with their wide diversity, the continuously increasing costs, the effect of deadlines extensions, … etc.

Due to the importance of finding a reliable estimate of the software development cost, this subject became the target of many research works. A close cooperation between the research and the industry was important in order

to develop and to validate estimation approaches. Cost estimation comprises the study of the three following attributes: the *effort*, the *time* and the *cost*.

Several techniques were proposed in order to formalize the cost estimation process [BOE81]. The techniques that are based on models are very well documented in the literature. These techniques can be classified into two categories [SHE97]:

- Parametric models
- Non parametric models

The parametric models are essentially based on a central equation expressing the effort as a function of certain attributes that affect the cost [HAL77] [WAL77] [PUT78] [BOE81] [BOE00]. These attributes are called *cost drivers*. Other parametric models are based on the calculation of function points [ALB83] [ABR96] [MAT94].

Non parametric models make use of artificial intelligence techniques such as analogy reasoning, genetic algorithms, rule based systems, and decision trees. These models were utilized to get rid of some of the inconveniences of parametric models. Non parametric models don't necessitate any specific form to express the cost in terms of cost factors. Examples of non parametric models are found in [HAM00], [POR90a], [POR90b], [MEN02], [SRI95], [SER95] [HUG96], [SHE97], [WIT97], [KAD00], and [BUR01].

Each type of the estimation models has its advantages and its disadvantages, and no one could prove that one type performs better than the other in all the situations. Actually the performance of a model depends on several characteristics of a given database of historical projects, such as their size, the number of attributes describing the projects, and the presence of extreme

points [SHE01]. The estimation using analogy and the Case-Based Reasoning (CBR) are promising techniques that are well adapted to the problem of cost estimation. These methods can be used when we have little knowledge about the problem to be solved, and for which the optimal solution is not known a priori. The estimation by analogy consists of using the of historical data to provide a solution of the studied problem [KAD00], [HAM00].

The thesis is composed of six chapters. In chapter 2 we present a survey on the different techniques used for software cost estimation. We emphasize more on the estimation techniques which are based on estimation models, being parametric or non-parametric. For each model we present the basic concepts, the advantages, the disadvantages, and some examples.

The third chapter presents the role of the fuzzy logic in the development of intelligent systems. This chapter discusses the advantages of using fuzzy logic for the presentation and the treatment of imprecision and the uncertainty in intelligent systems. It then presents the basic concepts of fuzzy logic such as the fuzzy sets, fuzzy propositions, fuzzy rules, fuzzy implication and aggregation of fuzzy rules.

In the fourth chapter, we discuss the massive use of linguistic variables by most of the cost estimation models. These models adopt a classic representation of these linguistic values. We illustrate the inconveniences of such representation, and we suggest the use of fuzzy representation. In this chapter we apply the fuzzy logic for the representation, and for the treatment of the six linguistic values used by the COCOMO model. We introduce new cost factors $Fuzzy\_C_{ij}$, which are calculated from the classical factors C_{ij} after modifying them using the membership functions associated with the different fuzzy sets.

In chapter 5 we evaluate the precision of the propose algorithm. The evaluation is made by comparing the estimations performed by the new approach with the results obtained from the original version of the intermediate COCOMO'81 model.

Finally in chapter 6, we present our conclusions and our suggestion for further studies.

Chapter 2

Software Cost Estimation Techniques

In this chapter, we give a literature review of the various cost estimation techniques in software development. We briefly present those techniques based on the *expertise*, the *analogy*, and the *descending* and *ascending* strategies. We then present in full details the technique based on modeling. This technique expresses the development effort as a function of the factors affecting the cost. For this technique there are two modeling approaches: the *parametric* approach, that uses statistical methods, and the *non-parametric* approach, that uses the artificial intelligence methods, such as the decision trees and the genetic algorithms. We terminate this chapter by summarizing the results obtained from some historic projects.

2.1. Man-Month measure

The development effort of software represents the number of persons as well as the number of months necessary to complete the project. For example, if 10 persons worked for 5 months in a project, the development effort of this project is then 50 Man-Months. Consequently, if we know the monthly salary of the persons working in the project, then the project cost is simply the sum of the salaries multiplied by the number of months of work. This implies that the three attributes: *effort*, *time* and *cost*, are closely dependent.

Actually the major part of the development costs corresponds to the salaries. Thus, the cost of a software project is directly proportional to the number of Man-Months necessary to finish of the project. There are obviously, other costs, for example the hardware, the transportation, and staff training, but those

are less difficult to obtain. These other costs are not based on the productivity of the personnel, or the software size to be developed.

Due to the fact that the software development cost is determined by the number of persons and months of work, i.e. the development effort, the two terminologies of *cost estimation* and *effort estimation* are considered equivalent in software engineering. In general, most of the approaches of software cost estimation determine, in a first phase, the development effort, and then, they evaluate the cost by using the salaries of the involved personnel. The Man-Month unit, often used to measure the effort of software development, was the subject of a relevant criticism in [BRO95]. He showed that assigning more programmers to a project running behind schedule will make it even later, due to the time required for the new programmers to learn about the project, and the increased communications overhead. When N people have to communicate among themselves (without a hierarchy), as N increases, their output decreases and can even become negative (i.e. the total work remaining at the end of a day is greater than the total work that had been remaining at the beginning of that day, such as when many bugs are created).

One of the results shown in [BRO95] was that persons and months are not interchangeable. It is not always true that we can reduce the development time by increasing the number of persons working on the project. This assumption remains valid in certain traditional industries where a task can be easily divided and implemented by a group of persons without intercommunication between them. Thus, in such cases, the addition of the personnel in the project makes it possible to reduce its development time. However, in software this situation is rarely encountered; the activities of software development are often

not divisible. Sometimes tasks can be partitioned but with very high rate of communication between the involved persons.

2.2. Cost estimation techniques

In the literature there are a plenty of cost estimation techniques. [BOE81] presents seven of these techniques that are widely used nowadays. These seven techniques are:

- Estimation based on expertise
- Estimation by analogy
- Estimation based on Parkinson law (a project will cost what you decided to spend)
- Estimation based on the customer price (a project will cost what the customer is willing to spend)
- The descending estimate (Top-down)
- The ascending estimate (Bottom-up)
- Estimation based on modeling

However, according to [FEN97] the techniques based on *Parkinson law* and the *customer price* cannot be considered as techniques for cost estimation. The method based on Parkinson law expresses the goal not an estimate. The technique based on Parkinson law is a constraint with the project which can determine its feasibility.

Each one of the above techniques has its advantages and disadvantages, but according to [BOE81] none of these techniques can be used exclusively. In the following we briefly present these techniques.