Empirical Estimation of COCOMO I and COCOMO II Using a Case Study

Muhammad M. Albakri[1]
M. Rizwan Jameel Qureshi[2]
[1-2]Department of Information Technology, King Abdul-Aziz University, P.O. BOX 80221 Jeddah 21589, Saudi Arabia

Abstract- *There are several software estimation models such as Line of Code, Function Point and COnstructive COst MOdel (COCOMO). The original COCOMO model is one of the most widely practiced and popular among the software development community because of its flexible usage. It is a suite of models i.e., COnstructive COst MOdel I and COnstructive COst MOdel II. In this paper, we are evaluating the both models, to find out the level of efficiency they present and how they can be tailored to the needs of modern software development projects. We are applying COCOMO models on a case study of an E-Commerce application, that is built using HTML and JavaScript. We will also shed light on the different components of each model, and how their Cost Drivers effect on the accuracy of cost estimations for software development projects.*

Keywords- *COCOMO I; COCOMO II; Software Cost Estimation; Software Cost Drivers' Assessment; Trade-off Analysis; Component Composition.*

1 Introduction

The main stimulus for the COCOMO I model is to help people understand the cost consequences of the decisions they will make in developing and supporting a software product. COCOMO II not only offers a cost estimation tool, but also provides a great amount of parameters which explain what the model is estimating, and why it produces the estimates it does. COCOMO I is actually a hierarchy of three sub-models and each sub-model is progressively more detailed than the other. This paper will present our results and findings after applying two of COCOMO's sub-models. The First sub-model is 'Basic COCOMO'. It is a single-valued model and calculates the software development cost and effort of a program by measuring lines of code (LOC). Basic COCOMO itself is divided into three modes based on the nature of the software project. First is 'Organic Basic COCOMO', it is used in small-sized simple software projects developed by small teams with good application experience. Second is 'Semidetached Basic COCOMO', it is used in medium-size software projects developed by teams with diversified levels of experience. Third is 'Embedded Basic COCOMO', that is used in massive software projects with strict resource constraints developed by multiple teams acquiring immense levels of experience, and sophistication. The second sub-model is 'Intermediate COCOMO'; it is simply 'Basic COCOMO' plus a set of subjective 'Cost Drivers'. Those drivers are used to assess product, computer, personnel, and project attributes of a software project. The evaluator uses a six-level scale to decide where each attribute fall. When an attribute is assessed, it produces what is called an Adjustment Factor. After all adjustment factors are multiplied together, they give an Effort Adjustment Factor (EAF) that is usually equal to a value between 0.9 and 1.4. The EAF is then mathematically applied on all Basic COCOMO's formulas. Third sub-model is Detailed COCOMO, as the name indicates, it produces the most accurate estimation of all three sub-models of COCOMO I. It combines Basic and Intermediate COCOMO together, boosted by an assessment of every Cost Driver's impact on each stage of 'Barry Boehm's software engineering process'. COCOMO II model on the other hand, is divided into four sub-models. Each sub-model is based on different inputs and estimates the effort of different activities of a software project. 'Application Composition' is the first sub-model. It estimates the effort of prototype systems developed using scripts, database programming, etc. And it uses application points as an input. Second sub-model is 'Early Design', it calculates initial effort based on system requirements and design options, and uses function points as an input. Third sub-model is 'Reuse', it estimates the effort of integrating reusable automatically generated components and uses generated line of code as an input. Fourth sub-model is 'Post Architectural', it estimates the development effort of system design specifications and uses lines of source code as an input.

The paper is further organized as: section 2 covers related work. Section 3 defines the research problem. Section 4 describes the brief case study design. Section 5 illustrates the evaluation. Section 6 covers the discussion.

2 Related Work

Boehm et al. [1] proposed evaluation criteria for the validity of the process models and they provided effective results. This article also explained the strengths and weaknesses of various cost estimation techniques for the period of 1965 to 2005 (40 years). COCOMO-II [2] was an excellent model up to 2005 but it did not enfold the new requirement and development styles for the reuseness or estimation of cost. COCOMO-II directed the software experts to create and designed new models such as the Chinese government version of COCOMO (COGOMO) and the Constructive Commercial-off-the-Shelf Cost Model (COCOTS) etc. Different future challenges were discussed for the invention of new model/methods and tools.

The author discussed different software cost estimation techniques and highlighted various hot areas and challenges of research in the field of software cost estimation. In [3], it is emphasized that there should be a need to research more in this field to open the new horizons for novice researchers. Nasir [4] discussed the strengths and weaknesses of various software estimation techniques to provide the basis for the exactness of software cost estimation. Basic Project Estimation Process also presented in a wonderful style. This paper clearly elaborated the different types of models those were derived from COCOMO (I&II).

Reusability of components in Component Based Development (CBD) is illustrated in [5]. The research in [5] also discussed and compared different architectures of CBD. It may be mentioned that a detail explanation of advantages and disadvantages of CBD elaborated very nicely. A comparison, of component based development (CBD) with other traditional software development practices, is also provided.

Succi and Baruchelli [6] highlighted the importance of standardization of components for the software reusability. The discussion of this research paper was to find how total development cost of a software system affected on the basis of component-based software engineering. The main two factors those were affecting the standardization cost of a component have been explained. According to them, the cost of the standardization of component(s) must be included during the cost-benefit analysis of a software system.

Gill [7] highlighted the pertinent issues of software reusability for component based development on the basis of CBSE, considered the important issues of software reusability and high level reusability guidelines. He mentioned that how much reusability resulted to improve product reliability and to reduce overall software development cost.

The problem of crosscutting which is produced during component development is elaborated in [8]. They solved this problem by the extension with Aspect oriented methodology. It was mentioned by an example that how new business rules resulted in the more adaptable and reusable components. Aspect Component Based Software Engineering has been developed with success in the CORBA Component Model domain [9].

Dolado [10] wrote a report for the validation of the component-based method (CBM) for software size estimation by the analysis of 46 projects. Then the complete process of this analysis and different techniques of analysis was mentioned. Relationship of LOC (Line of Code) and NOC (No. of Component) was carried out with suitable examples. Comparison of CBM and a Global Method (Mark II) was also provided [10].

3 Research Problem

A number of discussions are reported in the literature for the effectiveness of COCOMO models. This paper is written to find out the accuracy of cost estimation of both models when applied on a specific project. Further, what is the impact of cost drivers during the system development life cycle phases.

We want to validate the accuracy of the cost estimations of COCOMO models for projects that are built using HTML and JavaScript. Hence, we will not only find out how accurate and reliable they are, but also whether they are suitable for estimating HTML and JavaScript Code.

4 Case Study Design

The case study is for a project completed by author Mr. Albakri in a course called 'Human Computer Interaction'. The objective of the project is to follow the principles of HCI in creating an E-Commerce web application of an online bookstore. The application consists of forteen webpages written in HTML and JavaScript. All forteen pages were fully designed to have different content and perform different web tasks. Then, they were coded and connected together according to their design. The pages are a demo experience of how a real user would buy a book online. Detailes of the pages are mentioned in the next section.

5 Evaluation

The sub section 5.1 covers the COCOMO I whereas COCOMO II is covered in the sub section 5.2 subsequently.

5.1 Applying COCOMO I

Sub-model Used: Basic COCOMO I

Mode Used: Organic

Formulas Used:

$$Effort(Man\ Month) = 3.2 \times (KLOC)^{1.05} \quad (1)$$

$$Time = 2.5 \times (Effort)^{0.38} \quad (2)$$

Calculating Total LOC:

Table 1: HTML Pages, 14 pages in total:

Webpage Name	Number of Lines of Code
abousUs	101 LOC
bookDetails	119 LOC
categories	376 LOC
congrats	91 LOC
contactUs	144 LOC
feedBack	267 LOC
index	276 LOC
myAccount	114 LOC
payment	245 LOC
searchResults	327 LOC
shoppingCart	157 LOC
signIn	118 LOC
signUp	190 LOC
verification	146 LOC
	Total = 2918 LOC, 2.918 KLOC

Estimating Effort:

$$Effort = 3.2 \times (2.918)^{1.05}$$

$$Effort = 9.851\ MM$$

Estimating Time:

$$Time = 2.5 \times (9.851)^{0.38}$$

$$Time = 5.963\ Month$$

Sub-model Used: Intermediate COCOMO I

Mode Used: Organic

Formulas Used:

$$Effort(Man\ Month) = EAF \times 3.2 \times (KLOC)^{1.05} \quad (3)$$

$$Time = 2.5 \times (Effort)^{0.38} \quad (4)$$

Cost Drivers:

- Product Attributes:
  1. RELY- Required Software Reliability.
  2. DATA – Database Size.
  3. CPLX – Product Complexity.
- Computer Attributes:
  4. TIME – Execution Time.
  5. STOR – Main Storage.
  6. VIRT – Virtual Machine Volatility.
  7. TURN – Computer Turnaround Time.
- Personal Attributes:
  8. ACAP – Analyst Capability.
  9. AEXP – Applications Experience.
  10. PCAP – Programmers Capability.
  11. VEXP – Virtual Machine Experience.
  12. LEXP – Programming Language Experience.
- Project Attributes:
  13. MODP – Use of Modern Programming Practices.
  14. TOOL – Use of Software Tool.
  15. SCED – Required Development Schedule.

Table 2: Estimating Cost Drivers Values:

	Very Low	Low	Normal	High	Very High
RELY				1.15	
DATA					1.16
CPLX					1.30
TIME		0.85			
STOR				1.21	
VIRT				1.30	
TURN				1.15	
ACAP				0.86	
AEXP		0.80			
DCAP			1.0		
VEXP				0.90	
LEXP				0.95	
MODP			1.0		
TOOL					0.83
SCED		0.85			

Calculating Effort Adjustment Factor(EAF):

Here all assessment values are multiplied together to determine the EAF:

$$EAF = 1.15 \times 1.16 \times 1.30 \times 0.85 \times 1.21 \times 1.30 \times 1.15 \times 0.86 \times 0.80 \times 1.0 \times 0.90 \times 0.95 \times 1.0 \times 0.83 \times 0.85 \quad (5)$$

$$EAF = 1.1$$

The equation further substitutes as follows.

$$Effort(Man\ Month) = 1.1 \times 3.2 \times (2.918)^{1.05}$$

$$Effort = 10.836\ MM$$

$$Time = 2.5 \times (10.836)^{0.38}$$

$$Time = 6.182\ Month$$

Sub-model: Organic Detailed COCOMO I

This sub-model was not used this model based upon two reasons. First, is that this E-Commerce application does not require to go through the detailed project phases of 'Barry's software engineering process. Second, it is a

small scale project. Though, the findings will be stated in section 6.

5.2 Applying COCOMO II

Sub-model Used: Application Composition

Formulas Used:

$$Object\ Point(OP) = Estimated\ Count(EC) \times Complexity\ Factors(CF) \quad (6)$$

$$New\ Object\ Point(NOP) = OP \times \frac{(100-\%reuse)}{100} \quad (7)$$

$$Effort(Man\ Month) = \frac{NOP}{Productivity} \quad (8)$$

$$Cost/NOP = \frac{Labor\ Rate}{Productivity} \quad (9)$$

$$Total\ Project\ Cost = \frac{Cost}{NOP} \times NOP\ of\ current\ project \quad (10)$$

Table 3: Project Parameters:

		CF		
	EC	Simple	Average	Complex
Screens	14	1	2	3
Reports	6	2	5	8
Components	10	1	1	10
Environment Maturity	25			

After Substitution:

$$Object\ Point(OP) = (14 \times 2) + (6 \times 5) + (10 \times 1) \quad (11)$$

$$OP = 68\ OP$$

This e-commerce application reuses 90% of previous common e-commerce applications.

$$New\ Object\ Point(NOP) = 68 \times \frac{(100-90)}{100}$$

$$NOP = 6.8 \approx 7\ NOP$$

$$Effort(Man\ Month) = \frac{6.8}{25}$$

$$Effort = 0.272 \approx 1\ Man\ Month$$

Assumed, the average labor rate is 1500 USD.

$$Cost/NOP = \frac{1500}{25}$$

$$Cost/NOP = \$60$$

$$Total\ Project\ Cost = 60 \times 6.8$$

$$Total\ Project\ Cost = \$408$$

Sub-model Used: Early Design

Formula Used: $Effort = A \times Size^{B} \times M$ (12)

Where:

A is 2.94, a coefficient proposed by Boehm,

Size, is in KLOC,

B, reflects the increased effort required as the size of the project increases, ranges from 1.1 to 1.24.

M, is a multiplier which is based on a simplified set of seven project characteristics that influence the estimation.

Project Characteristics:

1. RCPX – Product Reliability and Complexity
2. RUSE – Reuse Required
3. PDIF – Platform Difficulty
4. PERS – Personnel Capability
5. PREX – Personnel Experience
6. SCED – Schedule
7. FCIL – Support Facilities

Table 4: Estimated Project Characteristics:

	Very Low	Low	Normal	High	Very High
RCPX				1.5	
RUSE					1.40
PDIF				1.20	
PERS			1		
PREX			1		
SCED		0.85			
FCIL			1		

Calculating multiplier M:

$$M = 1.5 \times 1.40 \times 1.20 \times 1 \times 1 \times 0.85 \times 1$$

$$M = 1.6$$

B value is medium (equal to 1.15), because the size of this e-commerce application is predicted to require medium expansion effort.

$$Effort(Man\ Month) = 2.94 \times 2.918^{1.15} \times 1.6$$

$$Effort = 16.118\ Man\ Month$$

Sub-model Used: Reuse

Formula: $Effort\ (Man\ Month)_{Auto} = \frac{ASLOC \times \frac{AT}{100}}{ATPROD}$ (13)

This formula estimates generated code. Where:

'Auto', indicates that 'Effort' is of generated code,

ASLOC, is the number of adaptive LOC of reusable components,

AT, is the percentage of adapted generated code,

ATPROD, productivity of engineers integrating the code, usually approximates to 2400 LOC/Month.

Here, it is assumed that the HTML and JavaScript code of reusable components is generated using design models inserted into a code generator.

$$Effort\ (Man\ Month)_{Auto} = \frac{291.8 \times \frac{0.1}{100}}{2400}$$

$$Effort_{Auto} = 0.000121\ Man\ Month$$

Sub-model Used: Post-Architectural

Formula: $Effort = A \times Size^{B} \times M$ (14)

As the name indicates, this model is used when more project parameters become identified.

Detailed Project Cost Drivers:

1. RELY – Required Reliability.
2. CPLX – Complexity of Modules.
3. DOCU – Extent of Documentation.
4. DATA – Database Size.
5. RUSE – Required percentage of Reusable Components.
6. TIME – Execution Time Constraint.
7. PVOL – Platform Volatility.
8. STOR – Memory Constraints.
9. ACAP – Analysts Capability.
10. PCON – Personal Continuity.
11. PCAP – Programmer Capability.
12. PEXP – Programmer Experience.
13. AEXP - Analyst Experience.
14. LTEX – Language and Tool Experience.
15. TOOL – Use of Software Tools.
16. SCED – Development Schedule Compression.
17. SITE – Extent of Multisite Working and Quality of Inter-Site Communication.

The last cost driver 'SITE' was excluded because the work site is not relevant to the nature of this application.

Table 5: Estimated Cost Drivers:

	Very Low	Low	Normal	High	Very High
RELY				1.15	
CPLX					1.30
DOCU				1.1	
DATA					1.16
RUSE					1.40
TIME		0.85			
PVOL		0.70			
STOR				1.21	
ACAP				0.86	
PCON				0.90	
PCAP			1		
PEXP			1		
AEXP		0.80			
LTEX			1		
TOOL					0.83
SCED	0.85				

$$M = 1.15 \times 1.30 \times 1.1 \times 1.16 \times 1.40 \times 0.85 \times 0.70 \times 1.21 \times 0.86 \times 0.90 \times 1 \times 1 \times 0.80 \times 1 \times 0.83 \times 0.85$$

$$M = 0.84$$

$$Effort\ (Man\ Month) = 2.94 \times 2.918^{1.15} \times 0.84$$

$$Effort = 8.462\ Man\ Month$$

6 Discussion

As mentioned above in Table 2, the estimated fifteen cost drivers provide more information about different areas of the application that were not obtainable at the beginning of the project to enhance cost and effort estimations. In the basic stage, the effort and time to develop the application was 9.851 MM and 5.963 Months respectively, but when intermediate stage was reached, 0.985 more effort and 0.219 more time was needed to complete the project.

At the beginning of the project, 'Organic Basic COCOMO' envisions the system as a single unit, whereas 'Organic Intermediate COCOMO' divides the system into subsystems or components. The intermediate cost drivers allow estimating particular components not the entire system, therefore enabling the development team to choose the best course of action regarding project's plan.

In 'Detailed COCOMO' the estimator's understanding does not only cover different project parameters, it also considers the project as a sequence of phases and each phase is estimated in a different way. That is the most major difference between it and previous sub-models. The 'Detailed COCOMO' assigns different cost drivers for each

of phase of the project. These phase-dependant cost drivers are the reason behind producing much more accurate estimations. For example if we consider the 'ACAP' cost driver, it is assigned a value of 1.00 for Coding and Unit Testing phase, which has no influence on the multiplication operation, but a value of 1.40 for Requirements phase. This intelligent manipulation of cost drivers can save up analysts' energy and time for phases that need them. They do not have an impact on coding and testing phases, because they are not involved.

COCOMO I only uses number of thousands lines of code (KLOC) as an input, and so it is best used in projects built using structured programming languages.

'Application Composition' is intended for prototype projects, where the project is built by composing components called 'Object Points'. It not only considers cost drivers but also project environment's characteristics like developer's experience and capability, CASE tool's maturity and capability, number of screens, and number of system generated reports. So, each component (object) is customized and then attached to whole body of the project in a different way and with a different level of challenge. As mentioned in previous section, 90% of the project is reused components from existing e-commerce applications. The relation between '%reuse' and 'Effort' is disproportional; the more components are reused (90% out of 100%), the less effort (1 Man Month) is required.

'Early Design' model is similar to 'Organic Intermediate COCOMO'. In each model more information is uncovered as initial stages of the project are concluded and design stages are initiated. However, only a rough system design is required to make early estimations. A very important element in the formula is 'B' as mentioned in section 5.2.2, which has a great influence on the effort estimation. While comparing the value of effort estimated by 'Application Composition' and the value of effort estimated by 'Early Design', the impact of 'B' is definitely obvious. More 15.118 man effort is needed to meet the increasing size of the project.

When 'Reuse' model is used HTML code can be generated using various code generators. It may appear that 'Early Design' model and 'Reuse' model are similar, because they both estimate reusable components, but that is not the case. In fact, the estimator using 'Early Design' needs neither to understand the reusable components nor to modify it. The estimator simply just uses them. Each time the generated code is studied and then refined, the cost of the component will decrease.

'Post-Architectural' model is used once an initial architectural design of the system is available. The model uses the same formula as 'Reuse' model uses. However, the estimation is the most accurate and realistic among others, because ten more cost drivers are uncovered and used in the formula. In section 5.2.2, 16.118 man effort is needed to develop the e-commerce application. This estimation was found to be unrealistic, because conventionally such a small scale application (2918 LOC) does not need that amount of man effort to be completed. Thanks to the detailed cost drivers, effort is reduced to 8.462 Man Month.

Acknowledgements

The author Muhammad Albakri would like to thank Dr. M. Rizwan Jameel Qureshi for his continuous and valuable support. It would never been possible to complete the work without his guidance throughout the making of this paper.

7 References

[1] Boehm, B. W. and R. Valerdi. Achievements and Challenges in Cocomo-Based Software Resource Estimation published by IEEE Computer Society. 74-83 (2008).

[2] Boehm, B. W. An Overview of the COCOMO 2.0 Software Cost Model (1999).

[3] Zaid, A., M. H. Selamat, A. A. A. Ghani, R. Atan and K. T. Wei. Issues in Software Cost Estimation, IJCSNS Int J of Computer Science and Network Security, 8(11): 350-356 (2008).

[4] Nasir, M. A Survey of Software Estimation Techniques and Project Planning Practices, Proceedings of the Seventh ACIS Int. Conf. on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'06), (2006).

[5] Qureshi, M. R. J. and S. A. Hussain. A Reusable Software Component-Based Development Process Model Int. J of Advances in Engineering Software, 39(2): 88-94 (2008).

[6] Succi, G. and F. Baruchelli. The Cost of Standardizing Components for Software Reuse, Standard View 5(2) (1997).

[7] Gill, N. S. Reusability Issues in Component-Based Development, ACM SIGSOFT Software Engineering Notes, 28(4): 4 – 4 (2003).

[8] Clemente, P. J. and J. Hernández. Aspect Component Based Software Engineering, University Extremadura. 1-4 Spain (2001).

[9] Frakes, W. B. and K. Kang. Software Reuse Research: Status and Future, IEEE Transactions on Software Engineering, 31(7): 529-536 (2005).

[10] Dolado, J. J. A Validation of the Component-Based Method for Software Size Estimation, IEEE Transactions on Software Engineering, 26(10): 1006-1021 (2000).