International Journal of Recent Research and Review, Vol. VIII, Issue2, June 2015

ISSN 2277 – 8322

Problems with Cocomo-11 in Respect to Domestic IT Companies

Rekha Verma[1], Meenu Dave[2]

Department of Computer Science, Jagan Nath University, Jaipur, India

Email: [1]rekhaverma51@gmail.com

Abstract- **COCOMO II is** a **cost estimation model for planning and estimating software projects. It is an important aspect for managing software projects or software development business. A cost model provides a framework for communicating business decisions among the stakeholders of a software effort. COCOMO II helps in taking decision for price negotiations, improvement process, purchase of tools, architecture changes, make/buy tradeoffs, and several other return-on investment decisions with a reliable basis of estimate. COCOMO II** works effectively **for large and very large projects, but it seems that there are lots of short comings in** the **model** when applied to **small Indian domestic market. This paper introduces some of the problems of COCOMO II in relation of Indian domestic projects. A few solutions are** also **suggested here.**

Keywords- **COCOMO-II, effort multipliers, scale factors.**

I. INTRODUCTION

COCOMO (Constructive Cost Model) is a method that allows software project managers to estimate project cost and duration. It was developed initially (COCOMO 81) by Barry Boehm in the early eighties. The COCOMO II model is update of COCOMO 81 to address new practices of software development during 1990's and 2000's. The model is by now invigorative software engineering artifact that has, from customer perspective, the following features [1, 2]:

- The model is simple and well tested
- Provides about 20% cost and 70% time estimate accuracy

II. COCOMO II: FEATURES

COCOMO II incorporates various sub-models that produce the following detailed software estimates [3,4]:

A. *Estimation Equations*

The amount of effort in person-months (PM) is estimated by the formula:

$$PM = A \text{ x } Size^{E} \text{ x } \prod_{i=1}^{n} EM$$

The value of n is 16 for the Post-Architecture model effort multipliers, Em_i, and 6 for the Early Design model; the number of SF_i stands for exponential scale factors.

The values of A, B, SF_1 …, and SF_5 for the early design model are the same as those for the Post-Architecture model. The values of EM_1, …, and EM_6 for the early design model are obtained by combining the values of their 16 Post-Architecture counterparts.

The value of *A* and *B* in the COCOMO II are:

A = 2.94 B = 0.91

$$E = B + 0.01 \times \sum_{j=1}^{5} SF_j$$

B. *Scale Factors*

- Precendentedness (Prec)
- Development Flexibility (Flex)
- Architecture/Risk Resolution (Resl)
- Team Cohesion (Team)
- Process Maturity (Pmat)

C. *Effort Multipliers*

a) *General Cost Drivers*

- Required Software Reliability (RELY)
- Database Size (DATA)
- Product Complexity (CPLX)
- Development for Reusability (RUSE)
- Documentation (DOCU)

b) *Platform Factors*

- Execution Time Constraint (TIME)
- Main Storage Constraint (STOR)
- Platform Volatility (PVOL)

c) *Personnel Factors*

- Analyst Capability (ACAP)
- Programmer Capability (PCAP)
- Personnel Continuity (PCON)
- Applications Experience (APEX)
- Platform Experience (PLEX)
- Language and Tool Experience (LTEX)

d) *Project Factors*

- Use of Software Tools (TOOL)
- Multi site Development (SITE)
- Required Development Schedule (SCED)

III. DOMESTIC INDIAN MARKET: OBSTACLES

There are lots of problems which are being faced by the Indian IT companies who are working in the domestic market. Some of the important issues are:

A. *Small Budget*

Most of the domestic projects maintain a very low budget, which is either in a few thousands or a few lacs. Thus, maintaining good quality is not possible. Market competition and several other expectations put immense pressure on the software professionals and the provided budget is generally insufficient.

B. *Late Release of Payments*

The scenario for payment in the Indian market is not very encouraging for the small companies. As the business clients do not always pay on time, the companies work under constant pressure. A complete cycle of late payments start as the clients pay late, and thus the company also delays the payments to the software professionals, industry related vendors, and so on.

C. *Additional Manpower (software professionals) not available*

Small companies cannot afford bench of employees. Thus with a limited team of professionals, a company cannot take varied size projects. In this way, a company sometimes loses big project offers. Working with multiple projects is also not possible in this scenario.

D. *Work overload results in decreased work efficiency and low quality end product*

Every phase for a software development should be critically carried out. In big companies, the professionals are specifically assigned to phases like requirement analysis, designing, coding, testing, and maintenance, etc. Regularly working for increasing the quality output for a particular phase increases the efficiency and reduces the output time for that phase. On the other hand, a small company has limited professionals, who are always overburdened and the same person is responsible to take care of different phases of software development. Work overload and juggling between varied responsibilities results in reduced quality of the end product and decreased efficiency of the software professional.

E. *Frequent communication with the client is not possible*

Effective communication with the client lays foundation for a good quality end product. Large enterprises have a dedicated team which interacts with the clients regularly and conveys feedback to the development team. Small companies do not have such kind of setup. Mainly it is a team of 2-3 professionals, who are themselves interacting with the client, and taking care of the complete SDLC also. Thus, because of work pressure, there is minimal interaction with the client and feedback based development or rectification is also not carried out frequently.

F. No Good Methodology for Cost Estimation

Most of the methodologies of cost estimations are using US standard of working. This methodology doesn't get fit in Indian market/domestic market. Using of these techniques are very time consuming and complex in use. For example, for using COCOMO-11, expert knowledge of COCOMO-11 is required.

IV. SUGGESTIVE MEASURES IN REFERENCE TO DOMESTIC INDIAN MARKET

Keeping in mind the aforesaid problems, COCOMO-II has been simplified for domestic Indian market in the following ways:

- By removing unnecessary scale factors and effort multipliers.
- By designing and developing a java based tool with modified parameters for COCOMO-II.

A. Removal of unnecessary scale factors and effort multipliers

COCOMO-II is based on 5 scale factors and 16 effort multipliers. By thorough study and observations, it is found that some of them are not required for small domestic projects. Following scale factors were removed from COCOMO-II for specific use in the domestic applications:

a) Development Flexibility (Flex)

The FLEX scale factor [5] is largely intrinsic to a project and uncontrollable. For small domestic project FLEX does not play important role so this can be avoided.

b) Architecture/Risk Resolution (RESL)

This factor [5] combines two of the scale factors in COCOMO, Design Thoroughness by Product design Review (PDR) and Risk Elimination by PDR. RISK Resolution is not much important for domestic projects so this can also be ousted.

c) Team Cohesion (TEAM)

The Team Cohesion scale factor [5] accounts for the sources of project turbulence and entropy because of difficulties in synchronizing the project's stakeholders: users, customers, developers, maintainers, etc. These difficulties may arise from differences in stakeholder objectives and cultures, difficulties in reconciling objectives, stakeholders' lack of experience and familiarity in operating as a team. In domestic projects, a very less number of stakeholders are available; so this is not a important factor and can be ignored.

A few of effort multipliers mentioned below have also been removed from the implementation.

a) Database Size (DATA)

This cost driver [6] attempts to capture the effect that large test data requirements have on product development. The rating is determined by calculating D/P, the ratio of bytes in the testing database to SLOC in the program. The reason the size of the database is important to consider is because of the effort required to generate the test data that will be used to exercise the program. In other words, DATA is capturing the effort needed to assemble and maintain the data required to complete test of the program through IOC. As most of the domestic projects are having quite less number of tables and work with average size of databases, so this effort multiplier can be avoided and there is no need to consider it while working with domestic small projects.

b) *Development for Reusability (RUSE)*

This cost driver [5] accounts for the additional effort needed to construct components intended for reuse on current or future projects. This effort is consumed with creating more generic design of software, more elaborate documentation, and more extensive testing to ensure components are ready for use in other applications. "Across project" could apply to reuse across the modules in a single financial application project. "Across program" could apply to reuse across multiple financial applications projects for a single organization. "Across product line" could apply if the reuse is extended across multiple organizations. "Across multiple product lines" could apply to reuse across financial, sales and marketing product lines. Most of the domestic projects are small and cannot be sub-divided in reusable components. So this effort multiplier need not be considered for domestic projects.

c) Documentation (DOCU)

Several software cost models have a cost driver for the level of required documentation [3], i.e., Documentation Match to Life-Cycle Needs (DOCU). In COCOMO II, the rating scale for the DOCU cost driver is evaluated in terms of the suitability of the project's documentation to its life cycle needs. In general, domestic projects do not use formal documentation; therefore, this multiplier can be removed.

d) Platform Volatility (PVOL)

"Platform" is used here [3] to refer to the complexity of hardware and software (OS, DBMS, etc.), the software product calls on to perform its tasks. If the software to be developed is an operating system, then the platform is the computer hardware. If a database management system is to be developed, then the platform is the hardware and the operating system. If a network text browser is to be developed, then the platform is the network, computer hardware, the operating system, and the distributed information repositories. The platform includes any compilers or assemblers supporting the development of the software system. This rating ranges from low (where there is a major change every twelve months), to very high (where there is a major change every two weeks). Domestic projects are generally developed for single platform only, hence refrain from using this multiplier is suggested.

e) Personnel Continuity (PCON)

The rating scale for PCON [3] is in terms of the project's annual personnel turnover: from 3 percent, i.e. very high continuity, to 48 percent, which is very low continuity. Most of the domestic projects are of small time duration so personnel continuity does not play much role over here.

f) Multi site Development (SITE)

Given the increasing frequency of multisite development, and indications that multisite development effects are significant, the SITE cost driver [3] has been added in COCOMO II. Determining its cost driver rating involves assessing and judgment-based averaging of two factors: site collocation from surface mail and some phone access to full interactive multimedia. Domestic projects are generally being developed on a single site, so, multi-site development aspect may not be considered.

B. Design and Development of Java Based Tool

For fast estimation of cost, a Java based tool has been designed by removing unnecessary scale factors and effort multipliers form the original COCOMO-II model.

V. ANALYSIS OF THE MODIFIED TOOL FOR COCOMO-II

After removing some of the effort multipliers and scale factors, a few projects were evaluated against both the original COCOMO-II and the modified tool. The results achieved are shown in Table 1 shown below.

TABLE I
Effort Estimation in person-months (PM)

Projects	**Value Estimated by COCOMO II**	**Value Estimated by Modified tool**	**Actual Value**
P1	5	4.6	4
P2	13	11.83	10
P3	7	5.76	5
P4	1.2	0.76	0.53

The tabular values when converted in graphical form, as given in Figure 1, clearly show that most of the values estimated by the modified tool are closer to the actual values.

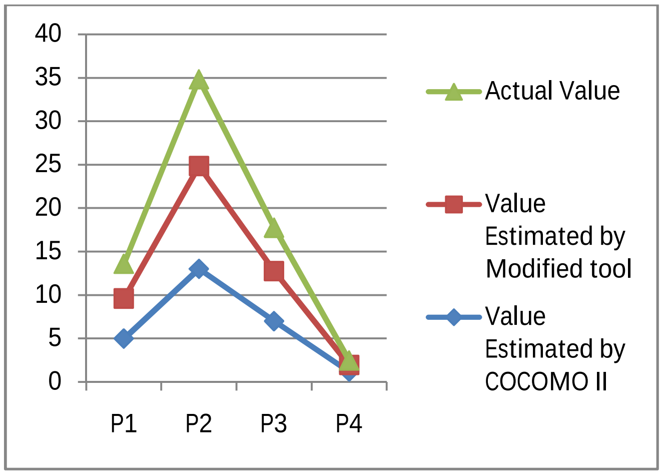

Fig.1 Effort estimation for domestic indian market projects

VI. CONCLUSION

COCOMO-II cannot be applied effectively on the projects for the domestic Indian market, as many effort multipliers and scale factors are not relevant for small/medium projects. The paper is based on study of lots of Indian projects meant for the domestic market. The modified COCOMO-II tool based on the adaptations carried out as a result of the study has proved that with changes, COCOMO-II will prove to be more effective for the domestic Indian market.

VII. REFERENCES

[1] Pressman, Roger S. (2005). Software Engineering, A Practitioner's approach, McGraw-Hill International Edition (6th edition), pp. 674-703.

[2] Center for Software Engineering (2000). COCOMO II - Model Definition Manual, University of Southern California (USC), version 2.1. Available online at: http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf

[3] Khatibi, V. and Jawawi, D.N.A. (2011). Software Cost Estimation Methods: A Review, Journal of Emerging Trends in Computing and Information Sciences, Volume 2 No. 1, pp. 21-29.

[4] Boehm, B., et al. (2005), COCOMO Suite Methodology and Evolution, CrossTalk - The Journal of Defense Software Enfineering, pp. 20-25. Available online at: http://sunset.usc.edu/csse/TECHRPTS/2005/usccse2005-509/usccse2005-509.pdf

[5] Merlo–Schett, N. (2003), COCOMO, Seminar on Software Cost Estimation, Requirements Engineering Research Group, Department of Computer Science, University of Zurich, Switzerland, pp. 1-20.

[6] De Rore, L., et al. (2008),Deducing software process improvement areas from a COCOMO II-based productivity measurement, Proceedings of 5th Software Measurement European Forum, Milan, May 28-30 2008, pp. 163-174. Available online at: http://www.dpo.it/smef2008/papers/proceedings_smef2008.pdf