THE COCOMO MODEL IN SOFTWARE COST ESTIMATION: A Review

Nivedita[1], Rachna Sharma[2]
[1]M. Tech (Scholar), [2]Assistant Professor
Department of computer science and Engineering, Baddi University of Emerging Sciences and Technology Baddi, Solan (India)

Abstract – Cost estimation is set of methods to expect genuine cost essential for software development. Since many years software engineers have tried various process for cost estimation, which helps them to reason schedule implications of development, investment decisions. In this paper, we have discussed the importance of cost estimation, software architecture and architectural goals. In addition, we reviewed software development life cycle, different phases of SDLC and different SDLC based models like: waterfall, spiral, agile model etc. Furthermore, we compared the SDLC models in terms of their benefits and limitations. Finally, we've discussed COCOMO model and levels of COCOMO and several issue in COCOMO model. Previous research in this area was surveyed to get better idea of cost estimation.

Keywords – *COCOMO, Software Development life Cycle, Software architecture and SDLC models.*

I. INTRODUCTION

Cost estimation incorporates the strategies that assist in anticipating the genuine and aggregate cost that will be required for our product and is considered as one of the perplexing and testing movement for the product organizations. They will probably create software, which is modest and simultaneously convey great quality. Programming cost estimation [1] is utilized fundamentally by framework investigators to get a guess of the basic assets required by a specific programming venture and their timetables.

1.1 Software Cost Estimation

Recently, Software is costlier product of computer frameworks projects. Huge part of cost in programming comprised of human effort and majority of cost estimation strategies focussed on this aspect and estimated as person months. Exact programming cost gauges are basic to the two engineers and clients. They can be utilized for creating demand for recommendations, contract transactions, booking, observing and control. Perfect cost estimation is important because:

- It can order and organize advancement ventures as for a general strategy for success.
- It can be utilized to figure out what assets to focus on project and accurate utilization of resources.
- It can be utilized to survey the effect of changes and support replanting.
- Tasks management is easier and controllable with assets and genuine requirements.
- Clients anticipate that real development expenses should match with estimated costs.

1.2 Software Architecture

Software is crucial structure of a software framework, discipline and documentation of those structures. These designs are expected reason about software framework.

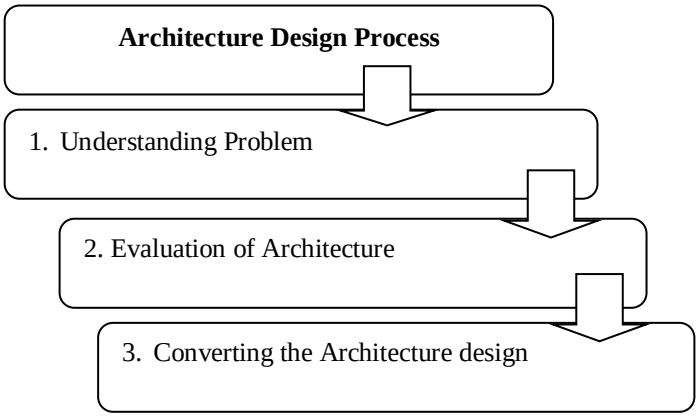

Figure 1.2 Architecture of Software

Every design includes programming components, relations between them and properties of two components and relations, [2] alongside basis for presentation and design of every component. The design of software framework is a similitude, practically equivalent to engineering of building. [3]

"The art or science of building: especially designing and building habital structures" [4]. Software design settles on basic auxiliary decisions that are expensive to change after implementation. Software engineering decisions, additionally named as compositional choices, incorporate particular basic choices from conceivable outcomes in programming outline. E.g., space shuttle launch vehicle controlling had prerequisite

of being quick and extremely dependable. Hence, a proper real-time processing dialect should be picked.

1.2.1 Architectural Goals

Application design seeks to construct an extension between business prerequisites and specialized necessities by understanding use cases, and later discovering approaches to actualize those utilization cases in the product. The objective of design is to recognize the prerequisites that influence the structure of application. Great engineering minimizes the business dangers related with building a specialized arrangement. A decent plan is adequately adaptable to have the capacity to deal with common float that will happen after some time in equipment and programming innovation, and in client situations and prerequisites. An engineer must think about general impact of plan choices, the inborn tradeoffs between quality characteristics, (like, security and execution), and tradeoffs required to address client, framework, and business prerequisites. Architecture must:

- Expose the structure of the framework however shroud the usage points of interest.
- Realize the majority of the utilization cases and situations.
- Try to address the prerequisites of different partners.
- Handle both useful and quality necessities.

1.3 SDLC

SDLC (Software Development Life Cycle), also known as software development process. Its a system that explains job execution at each step in software development process. An international standard for software life cycle process is ISO/IEC 12207, which aims to create high quality software that performs all tasks necessary for development and maintenance of software. The life cycle characterizes a technique for enhancing the nature of programming and the general advancement process.

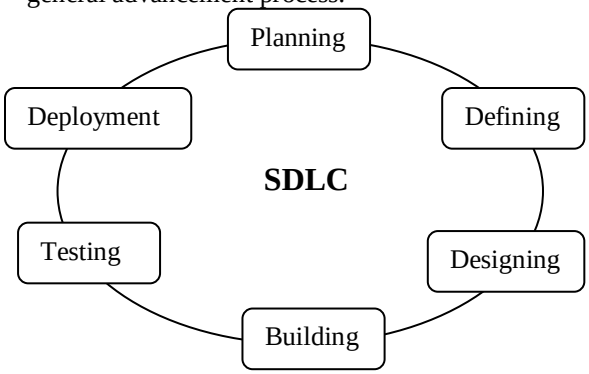

Figure 1.3 a standard SDLC.

1.3.1 Phases of SDLC

Stage 1: Planning and Requirement Analysis: Prerequisite investigation is principal step in SDLC. It is performed by senior individuals from the group with contributions from the client, the business division, advertise reviews and space specialists in the business. Planning for quality confirmation necessities and ID of risks related with the task is likewise done in the arranging stage. The result of specialized attainability is to characterize the different methodologies that can be taken after to execute the undertaking effectively with least dangers [5].

Stage 2: Defining Requirements: After analysing pre-requisites, next step is to define the requirements at different levels of software development and approving them from market analysts or client. This can be done with the help of SRS document.

Stage 3: Designing product architecture: In between the plan stage, engineers and specialized draftsmen begin the abnormal state outline of the product and framework to have the capacity to convey every prerequisite. The specialized subtle elements of the plan is talked about with the partners and different parameters, e.g., advances to be utilized, dangers, capacity of group, venture imperatives, time and spending plan are assessed and after that the best outline approach is chosen for the item.

Stages 4: Developing Product: The programming code is generated as per DDS during this stage. On the off chance that planning is performed in a point by point and composed way, code age can be expert without much issue. Engineers take after the coding rules characterized by their association and programming apparatuses like compilers, translators, debuggers and so on are utilized to create the code. Different programming dialects, for example, C, C++, Pascal, Java, and PHP are utilized for coding. The programming language is chosen w. r. t. type of software.

Stage 4: Testing: Testing is the last period of the Software Development Life Cycle before the product is conveyed to clients. In this stage we watch that our product is filling in according to our desire or not. We likewise check SRS that product full fill the whole necessity that said by the customer at the season of understanding.

Stage 5: Deployment and Maintenance: When programming is finished, the product can be deployed as per customer utilize and provide a specialized support group that care for any after generation issues. In the event that an issue is experienced in the generation the advancement group is educated and relying upon how serious the issue is, it may either require a hot-settle which is made and dispatched in a brief timeframe or if not exceptionally extreme.

1.4 SDLC Models

Several SDLC models are:

1.4.1 Waterfall Model

Waterfall is the conventional model of SDLC. In this model each stage is finished before going to next stage. There is no alternative for back-pedalling in the wake of moving to next stage. Waterfall is simple reasonable and easy to get it.

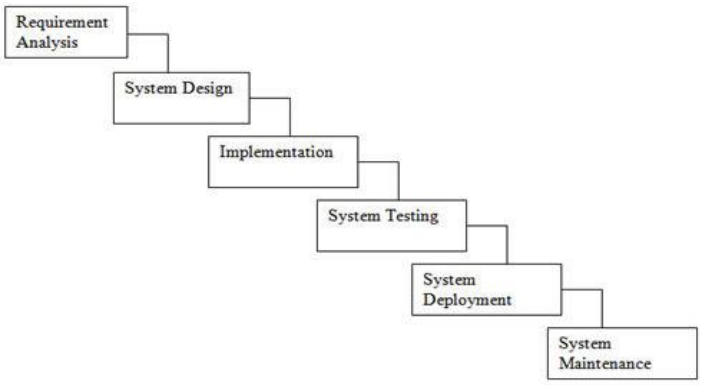

Figure 1.4.1 Waterfall Model [6]

1.4.2 V Model

V Model is advance waterfall display in which testing usefulness is included at each phase of the undertaking improvement rather than the venture fulfilment venture which prompts better task advancement. In this model likewise we can't move to following stage until or unless we can't finish the past advance.

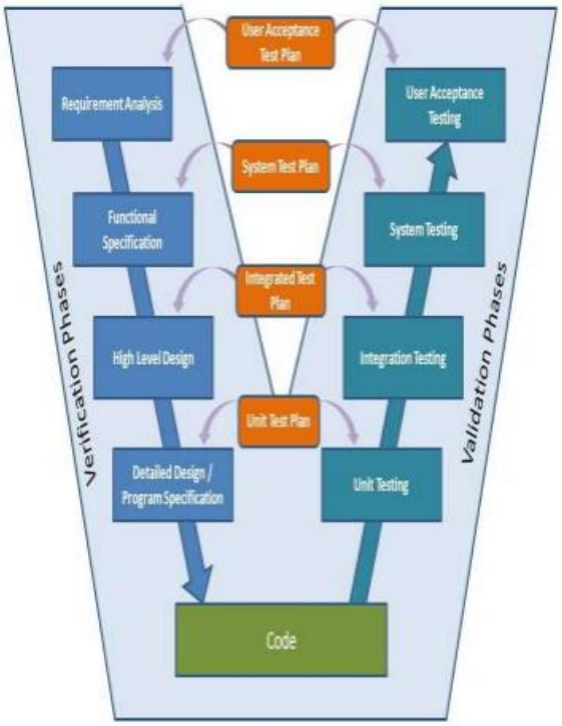

Figure 1.4.2 V Model [5]

1.4.3 Spiral Model

Spiral model is blend of the orderly and organized improvement which takes traits of emphasis Iterative model and furthermore joined these preferences with the effortlessness of the waterfall show with an extra overwhelming danger examination highlights.

Figure 1.4.3 Spiral Model [5]

Working of the Spiral model is isolated into four stages (recognizable proof, plan, assemble, assessment and hazard examination) and these four stages are get rehashed until the point that we won't get finish venture.

1.4.4 Agile Model

The light-footed model is half and half model it is utilizes points of interest of the both iterative and incremental model by separating programming item breaking an item into mechanical assembly where on each cycle or emphasis, a working model of a segment is conveyed. This model conveys refreshed discharges and each discharge contains some incremental updates and after consummation of every cycle item is tried to guarantee that the cycle is satisfactory or not

Figure 1.4.4 Agile Model [5]

Table 1: Comparison among SDLC Models

Model	Advantages	Disadvantages
Waterfall Model	• Easy to understand. • Prevents error propagation via verification and validation. • Well defined stages. • Less client involvement.	• Unable to go back to pervious phase.
Prototype Model	• Users/customers own requirements. • Instils customer confidence that the "right" product is being built.	• Lack of information about the exact number of iterations. • Premature prototypes lack key consideration like security, fault tolerance.
Spiral Model	• Less chances of failure. • Development can be terminated at any spiral, still working system is available.	• High risk analysis. • Suitable for bigger projects.
V – Model	• Easy to understand and implement • Quick error removal. • High success rate as compared to waterfall model.	• Not Flexible and rigid model. • High risks associated. • Goal is not clear.
Agile Model	• Adaptable to changes. • Focussed on client feedback. •	• Not feasible for complex projects. • Agile works well for small teams.

II. LITERATURE REVIEW

Shailendra Pratap Singh, et al., (2017) [7] proposed a new techniques are proposed to enhance the precision of cost estimation by altering parameters of COCOMO utilizing Homeostasis transformation based differential development (HMBDE). The basic concern in the field of programming advancement is estimation of the cost of programming at its underlying period of improvement. The cost estimation generally relies on the measure of the task, which may utilize lines of code or capacity focuses as measurements. In COCOMO, for the exactness of the cost estimation, cost factors should be planned in the individual improvement condition. The proposed strategy includes one more vector named as Homeostasis transformation vector in the current transformation vector to give more transfer speed to choosing viable mutant arrangements giving a wide pursuit space to likely arrangement. The proposed approach gives more exact answers for control the advancement. Execution of proposed calculation is contrasted and programming cost estimation models. The outcome checks that our proposed HMBDE performs superior to anything COCOMO based DE and PSO calculation and other delicate figuring models.

Alaa F. Sheta, et al., (2006) [8] presented two new model structures to gauge the exertion required for the advancement of programming ventures utilizing Genetic Algorithms (GAs). Characterizing the venture evaluated cost, term and support exertion ahead of schedule in the advancement life cycle is a profitable objective to be accomplished for programming ventures. Numerous model structures developed in the writing. These model structures consider displaying programming exertion as a component of the created line of code (DLOC). Building such a capacity encourages venture directors to precisely assign the accessible assets for the undertaking. A changed adaptation of the well-known COCOMO display gave to investigate the impact of the product improvement embraced strategy in exertion calculation. The execution of the created models were tried on NASA programming venture dataset. The created models could give a decent estimation capacities.

Wei Lin Du, et al., (2015) [9] proposed a hybrid intelligent model joining a neural system show coordinated with fluffy model (neuro-fluffy model) has been utilized to enhance the exactness of evaluating programming cost. Exact programming advancement exertion estimation is basic to the accomplishment of programming ventures. Albeit numerous strategies and algorithmic models have been created and actualized by experts, precise programming advancement exertion forecast is as yet a testing attempt in the field of programming building, particularly in dealing with unverifiable and loose information sources and collinear qualities. The execution of the proposed demonstrate is surveyed by planning and leading assessment with distributed task and mechanical information. Results have demonstrated that the proposed show exhibits the capacity of enhancing the estimation exactness by 18% in view of the Mean Magnitude of Relative Error (MMRE) measure.

D. Sivakumar, et al., (2017) [10] analyzed the COCOMO II show cost drivers and the effect of some predefined cost drivers in evaluating exertion and cost of programming ventures. The exact estimation technique for the most part depends on cost drivers in evaluating exertion and cost of programming ventures. The cost drivers and the determination of extents for a specific cost driver won't be same for all

models and circumstances. The assortment of cost drivers and its properties in the standard COCOMO II show in perspective of late situation is accomplished more spotlight on look into intrigue. This review ranges cost drivers and its esteems are balanced by the current modern circumstances and necessities. The quantity of cost drivers is decreased to 13 and the endeavors are evaluated utilizing this recently adjusted cost drivers. This model demonstrated its enhanced productivity in estimation with diminishment in level of MRE and MMR esteems.

Meiyappan Nagappan, et al., (2016) [11] examined momentum and future research slants inside the structure of the different stages in the product improvement life-cycle: prerequisites (counting non-useful), outline and advancement, testing, and support. There has been enormous development in the utilization of cell phones in the course of the most recent couple of years. This development has energized the advancement of a great many programming applications for these cell phones regularly called as 'applications'. Current appraisals show that there are a huge number of versatile application engineers. Subsequently, lately, there has been an expanding measure of programming designing examination led on portable applications to help such versatile application engineers. While there are a few non-practical prerequisites, we center on the subjects of vitality and security in our paper, since portable applications are not really worked by substantial organizations that can stand to get specialists for illuminating these two themes. For a similar reason we likewise examine the adapting parts of a portable application toward the finish of the paper. For every theme of intrigue, we first present the current advances done in these stages and afterward we introduce the difficulties show in ebb and flow work, trailed by the future openings and the dangers exhibit in seeking after such research.

III. COCOMO MODEL

Constructive Cost model was produced by Barry W Boehm in 1981. It is an algorithmic cost display. Algorithmic cost display is produced considering relating the present task to past undertakings. It relies upon recorded data. COCOMO relies upon size of the venture. The measure of the undertaking may change contingent on the capacity focuses [12].

COCOMOs are of 3 types:

- ***Basic COCOMO:*** Its used for relatively small projects. Little cost is included. Cost drivers mainly depend on size of project.
- ***Intermediate COCOMO:*** used for medium size project. Cost drivers are based on database size, execution, product reliability, etc.
- ***Advanced COCOMO:*** beneficial in bigger projects with large teams. Cost drivers based on analysis, requirement, design, testing and maintenance.

Table 2. Constant Value of COCOMO

	Constants			
Project modes	A	B	C	D
Organic project mode	2.4	1.05	2.5	0.38
Semi-detached project mode	3.0	1.12	2.5	0.35
Embedded project mode	3.6	1.20	2.5	0.32

COCOMO has been utilized seriously by programming chiefs and programming architects to help their product cost and estimation process because of the capacity to perform estimations with minimal master learning and experience.

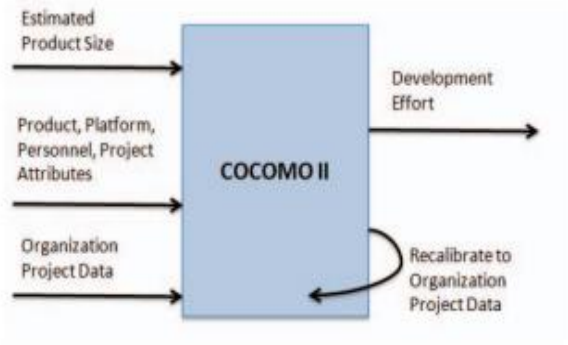

Figure 3.1 COCOMO Model [13]

The model has likewise helped programming supervisors and programming engineers in settling on basic advancement choices, for example, transactions on necessity changes, settling on compositional choices, perform chance administration choices or process change choices [13].

The major benefit of COCOMO Model is its simple to estimate cost whereas major drawback is that estimation in COCOMO Model is done at early stages of software development, which may lead to estimation failures.

IV. ISSUES IN COCOMO MODEL

COCOMO is cost estimation model of software development. The model utilize regression equation to estimation cost utilizing heuristic information with present and future qualities. COCOMO's estimation begins from the designing stage to combination period of cost and schedule of project. A separate estimation model ought to be required for residual stage. So COCOMO model isn't precise. A disentangled function point can be utilized for team and project size estimation. Such estimation is performed after creating design. Numerous endeavours and cost models depends on LOC, function point conversions are necessary. Requirement of

research data is less in contrast to LOC. Hence, function point is more accurate than COCOMO.

V. CONCLUSION

Cost estimation is set of techniques to expect bona fide cost basic for programming improvement. Since numerous years programming engineers have attempted different process for cost estimation, which encourages them to reason plan ramifications of advancement, speculation choices. In this paper, we have talked about the significance of cost estimation, programming engineering and compositional objectives. What's more, we explored programming advancement life cycle, distinctive periods of SDLC and diverse SDLC based models like: waterfall, winding, light-footed model and so forth. Moreover, we thought about the SDLC models as far as their advantages and impediments. At last, we've talked about COCOMO model and levels of COCOMO and a few issue in COCOMO demonstrate. Past research here was studied to show signs of improvement thought of cost estimation.

VI. REFERENCES

[1]. Pressman, R. S. (2005). *Software engineering: a practitioner's approach*. Palgrave Macmillan.

[2]. Clements, P., Garlan, D., Little, R., Nord, R., & Stafford, J. (2003, May). Documenting software architectures: views and beyond. In *Proceedings of the 25th International Conference on Software Engineering* (pp. 740-741). IEEE Computer Society.

[3]. Perry, D. E., & Wolf, A. L. (1992). Foundations for the study of software architecture. *ACM SIGSOFT Software engineering notes*, *17*(4), 40-52.

[4]. Merriam-Webster, Inc. (1983). *Webster's ninth new collegiate dictionary*. Merriam-Webster.

[5]. Rani, S. B. A. S. U. (2017). A detailed study of Software Development Life Cycle (SDLC) Models. *International Journal Of Engineering And Computer Science*, *6*(7).

[6]. Bassil, Y. (2012). A simulation model for the waterfall software development life cycle. *arXiv preprint arXiv:1205.6904*.

[7]. Singh, S. P., & Kumar, A. (2017, January). Software cost estimation using homeostasis mutation based differential evolution. In *Intelligent Systems and Control (ISCO), 2017 11th International Conference on* (pp. 173-181). IEEE.

[8]. Sheta, A. F. (2006). Estimation of the COCOMO model parameters using genetic algorithms for NASA software projects. *Journal of Computer Science*, *2*(2), 118-123.

[9]. Du, W. L., Capretz, L. F., Nassif, A. B., & Ho, D. (2015). A hybrid intelligent model for software cost estimation. *arXiv preprint arXiv:1512.00306*.

[10]. Sivakumar, D., & Janaki, K. (2017). Enhancing the Software Effort Prediction Accuracy Using Reduced Number of Cost Estimation Factors with Modified COCOMO II Model.

[11]. Nagappan, M., & Shihab, E. (2016, March). Future trends in software engineering research for mobile apps. In *Software analysis, evolution, and reengineering (SANER), 2016 IEEE 23rd International Conference on* (Vol. 5, pp. 21-32). IEEE.

[12]. Li, J., Ruhe, G., Al-Emran, A., & Richter, M. M. (2007). A flexible method for software effort estimation by analogy. *Empirical Software Engineering*, *12*(1), 65-106.

[13]. Boehm, B., Valerdi, R., Lane, J., & Brown, A. W. (2005). COCOMO suite methodology and evolution. *CrossTalk*, *18*(4), 20-25.