A Systematic Review of Software Maintenance Cost Estimation Methods

Chamkaur Singh[1], Dr. Neeraj Sharma[2], Dr. Narender Kumar [3]
[1]*Research scholar of IKG Punjab Technical University, Kapurthala, Punjab, India*
[2]*Professor, Gian Jyoti Group of Institutions, Mohali, India*
[3]*Assistant Professor, HNB Garhwal University,Srinagar Garhwal Uttarakhand*
(E-mail: dhillon.chamkaur@gmail.com)

***Abstract*—** Accurately estimating the cost of software projects is one of the most desired capabilities in software development organizations. Accurate cost estimates not only help the customer make successful investments but also assist the software project manager in coming up with appropriate plans for the project and making reasonable decisions during the project execution. Although there have been reports that software maintenance accounts for the majority of the software total cost, the software estimation research has focused considerably on new development and much less on maintenance. The paper provides an indication of the state of the art of software cost estimation (SCE). A suitable plan for maintenance action should be organized while emerging the software which is a significant feature of software conservation. This strategy should state the methods in which alterations are to be prepared. In this paper various factors of software maintenance and methods for cost estimation are discussed. The authors has defines various parameters of cost estimation and literature review related to this is also discussed. SCE models stands for software cost estimation models which helps in estimating the cost for software. In this paper various software estimation models such as COCOMO, Putnam, Estimacs, before you leap, price-s and SLIM are discussed in detail.

Keywords*—Cost Estimation, Cocomo, Accuracy, SLOC, Software Cost Estimation, Project control.*

I. INTRODUCTION

Maintenance of software may be a terribly broad activity that has enhancements in capabilities, error alteration, optimization and removal of obsolete competences. Modification is predictable and therefore, mechanisms should be developed for dominant, evaluating and creating deviations. So, any work done to amend the software package throughout its usage is taken into account to be maintenance work. The aim is to take care of the worth of software package over the amount. the worth are often improved by increasing the client base meeting, further needs and creating it a lot of economical cost by mistreatment newer technology [1]. Software package maintenance is a vital activity in software package engineering. Over the decades, software package maintenance prices are regularly reportable to account for a giant majority of software package prices. This reality isn't stunning. On the one hand, software package environments and needs square measure perpetually ever-changing, that cause new code upgrades to stay pace with the changes. On the opposite hand, the economic advantages of software package apply have inspired the software package trade to apply and enhance the present systems instead of to create new ones. Therefore, it's vital for project directors to predict and achieve the software package preservation prices effectively.

A suitable plan for maintenance action should be organized while emerging the software which is a significant feature of software conservation. This strategy should state the method in which alterations are to be prepared. The expensive to grow the software should comprise the price due to any essential alteration in the software. It means that maintenance charge is not only due to deprived design but also due to the alteration in client beliefs and environmental requirements in which the structure has been established. Additional, software maintenance is a strategy that comprises the range of maintenance, the maintenance individual /group and price estimation for software maintenance [3].

Variations in the software afterward it is distributed to the end employer originate the software maintenance cost. Software should be promoted according to the up gradation in the technique. Moreover, there may be interior concerns in software that needs repairs. Around 75 % of the total software growth charge is usually maintenance cost. Various factors of Software maintenance are discussed below:

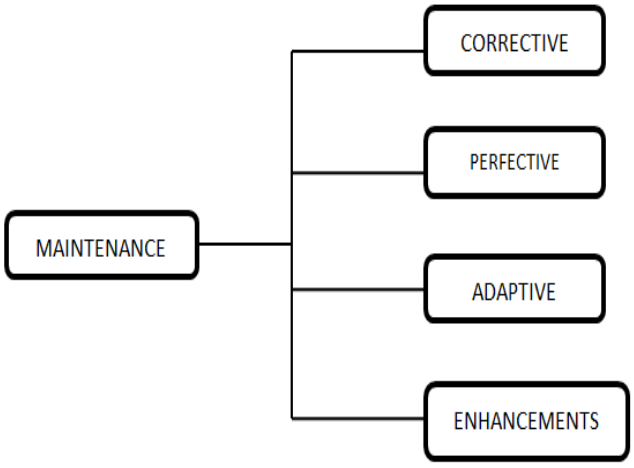

Fig 1. Software maintenance

Corrective maintenance – Nearly 20% of the price is due to fault alteration or problems in the software when it has been distributed client. Corrective maintenance means if software has been developed and client finds any errors or difficulties in the software it is the responsibility of software team to recover that error.

Adaptive maintenance – Almost 25 % of the cost is needed for the variations in the software to stay effective in a altering environment. It is most important for the soft wares that they should adapt every environment. Any software is called to be effective if it is able to adapt all types of environment. So if software is giving in any problem in adaptation it is the responsibility of team to recover it properly.

Perfective maintenance – Almost 5 % of the cost is needed to upgrading software to recover the performance. Software is effective only when it gives expected performance. Hence perfective maintenance is needed when any problem related to software's performance occurs.

Enhancements – Around 50 % of the cost required for the inventions to create the software up-to-date. If any updation is required in the software then it is software team's responsibility to do that thing for clients.

A. Software cost estimation methods and tools [9]:

Various methods for estimating software growth costs are available. Most of them are a mixture of the primary methods that are discussed below:

(1) Estimates made by an expert.

(2) Estimates based on reasoning by analogy.

(3) Estimates based on Price-to-Win.

(4) Estimates based on available capacity.

(5) Estimates based on the use of parametric models.

There are two main Software cost estimation methods that are discussed below:

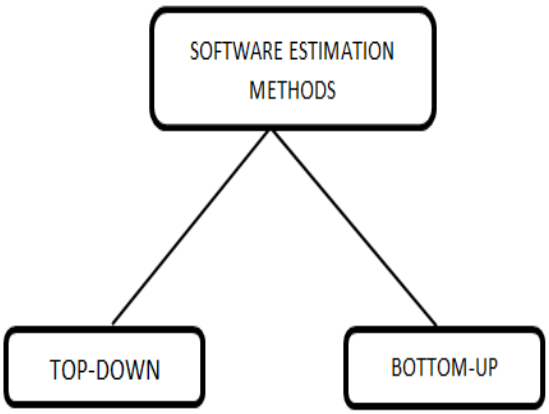

Fig 2. Software Estimation methods

(1) Top-down: In this method guesstimate of the complete project is derivative from the worldwide features of the product. The entire predictable budget is then divided amongst the several mechanisms.

(2) Bottom-up: In this the cost of every different module is predicted by the individual who will be liable for implementing the module. The distinct estimated budgets are summed to achieve the complete budget estimate of the scheme.

To compute the software maintenance price, many prototypes have been established and several of them are recycled by the corporations to compute the maintenance budget. The table below indicates the use of different techniques. The figs display that most of the companies use facts from previous plans in some way. Clearly this an casual way, since only 50% of the contributing groups record facts from finished projects. Approximations based on skillful judgment and the capability method proves to be reasonably popular regardless of the drawbacks of these approaches.

TABLE I. Use of cost estimation techniques (an organization can use more than one technique)

Methods	**Use (%)**
Expert judgment	25.5
Analogy method	60.8
Price-to-Win	8.9
Capacity problem	20.8
Parametric models	13.7

In the previous 3 periods, several software estimation prototypes and approaches have been suggested and recycled, such as COCOMO, SLIM, Price-S etc.

These are vital for software designers and their corporations, since it can afford cost mechanism, delivery accurateness, amongst many extra welfares for them. Numerous measureable prototypes of software cost estimation have been established and are created on the basis of size measure, like Line of Code (LOC) and Function Point (FP). It is clear that the correctness of size estimation straightly influence the correctness of cost estimation.

The size measures are discussed below

LOC: Source lines of code (SLOC), also called lines of code (LOC), is a software metric used to calculate the size of a computer program by calculating the total lines in the text of the program's source code.

Function point: It is a "unit of measurement" to state the quantity of business functionality an data system as a product delivers to a client. Function points are very important to calculate a functional size measurement (FSM) of the software.

II. WHAT MAKES SOFTWARE COST ESTIMATION SO DIFFICULT?

There are many causes that are listed below:

(1) There is a scarcity of knowledge on completed package comes. This sort of knowledge will support project management in creating estimates.

(2) Estimates are usually done in haste, while not associate degree appreciation for the hassle needed to try to a reputable job. Additionally, too usually it is the case that associate degree estimate is required before clear specifications of the system needs are created. Therefore, a typical state of affairs is that estimators are being pressured to put in writing associate degree estimate too quickly for a system that they are doing not absolutely perceive.

(3) Clear, complete and reliable specifications are troublesome to formulate, particularly at the beginning of a project. Changes, diversifications and additions are a lot of the rule than the exception: as a consequence plans and budgets should be custom-made too.

(4) Characteristics of package and package development create estimating troublesome. for instance, the amount of abstraction, complexity, quality of product and method, innovative aspects, etc.

(5) An excellent variety of things have associate degree influence on the hassle and time to develop package. These factors are referred to as 'cost drivers'. Examples are size and complexness of the package, commitment and participation of the user organization, expertise of the event team. normally these value drivers are troublesome to work out operative[10].

(6) Fast changes in data technology (IT) and therefore the methodology of package development are a tangle for a stabilization of the estimation method. for instance, it's troublesome to predict the influence of latest workbenches, fourth and fifth generation languages, prototyping methods, and so on.

(7) An associate degree reckoner (mostly the project manager) cannot have abundant expertise in developing estimates, particularly for big comes. what number 'large' comes will somebody manage in, for instance, 10 years?

A. Literature Review

Authors	Description
Boehm *et al.* (2000)	has tried to favorably estimate SLIM Model. SLIM is one of the widespread software maintenance budget estimation models and is extensively popular in the manufacturing for cost estimation. The study has exposed that SLIM method is relatively multipurpose and contributes precise outcomes in most of the circumstances. Though, it has been exposed that this method does not provide removed code as a maintenance activity. Henceforth, in this condition the outcomes attained via SLIM are not trustworthy.
Sneed et al. (2004)	Discussed a cost method for software maintenance and development depends on static and inconstant costs. The study has involved constraints which are derivative from fixed, variable flaws and output investigation of the software. Additionally, the varieties of jobs protected by this technique involves fault correction, practical improvement, monotonous modification and practical makeover. Though, the author has exposed that the projected method does not provide exact outcomes in case of recycling and repairing of web applications.
De Lucia *et al.* (2005)	Introduced an empirical study for structuring corrective maintenance effort estimation models. The study was supported variable regression toward the mean techniques. The authors prompt that completely different task sorts ought to be enclosed to enhance value estimation models. The task sorts prompt embody group A, B and kind C. group A considerations ASCII text file modification. B relates to fixing of knowledge misalignments and kind C considerations intervention not comprised within the previous classes. it's additional been projected that if task sorts for maintenance activities area unit troublesome for any project, then alternative models that area unit supported coarse-grained metrics ought to be used. Scope of this study is proscribed to corrective maintenance.
Nguyen et al. (2010)	has evaluated numerous fashionable package maintenance value estimation models and offered an extension of COCOMO II model. He has discovered that the varied existing models suffer from weaknesses regarding restricted selection of input metrics and restricted scope of maintenance activities. The author has evaluated and given the extended version of COCOMO II model for effort and size estimation of package maintenance comes. The author has used regression approach to create the estimation model. it's been emphasised that the planned model is used for the organizations wherever knowledge isn't ample to calibrate numerous estimation models. This extended version of COCOMO II model additionally considers SLOC metric of deleted code in its size metric. However, the author has cautioned that this model is proscribed to purposeful sweetening and fault correction activities of maintenance. Hence, the model needs further improvement to support reengineering, language and knowledge migration, performance improvement and alternative value effective activities.
Marounek et al.	presented a method for strength estimation in software maintenance. This method relies on an existing supposition

(2012)	by Jorgensen that around 83 to 84 %of all the estimation is completed by skillful prediction and estimation prototypes are not involved due to their complication. The suggested procedure is based on the addition of PERT formula around excellence of estimator and knowledge. Additionally, it is to be illustrious that this method relies on expert prediction. Therefore, the possibility of this method is also restricted.

III. SOFTWARE COST ESTIMATION MODELS

Before In this segment, one approximation method, specifically SCE prototypes, will be deliberated and the values of SCE prototypes defined, creating a difference among sizing and productivity models. The features of some famous models will also be specified.

A. The principles of SCE models

Models in these days are two-stage prototypes. The first phase is a size and another phase offers a output adjustment aspect. In the Initial phase an approximation about the size of the product to be established is found. In practice numerous sizing methods are included. The most famous sizes these days are function points and lines of code. On the other hand new sizing methods like 'software science 'and DeMarco's Bang technique have been described. The outcome of a sizing model is the size/volume of the software to be established, conveyed as the number of lines of source code, number of statements, or the number of functions points. In another phase it is predicted how much time and effort it will cost to cultivate the software of the expected size. Initially, the estimate of the size is transformed into an estimate in minimal man-months of effort. As this minimal effort have no benefit of information regarding the particular features of the software artifact, the method the software-product will be established and the production means, a number of cost manipulating factors (cost drivers) are included to the model. The outcome of these cost drivers must be predictable [11]. This outcome is frequently called a productivity adjustment factor. Some prototypes, like FPA, are focused more on the sizing phase. Others, like COCOMO model" it focus on the productivity phase and some apparatuses, such as Before You Leap conglomerate two prototypes to cover both phases. Fig 3 displays the two phases in SCE models.

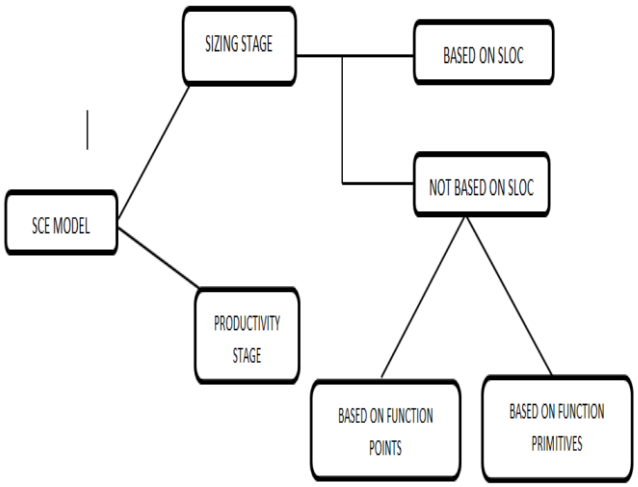

Fig 3. Cost Estimation Phases

B. An overview of SCE models

In the previous 10 centuries a amount of SCE prototypes have been established. This segment does not provide an exhaustive management of all the models: the overview is restricted to one instance of a sizing prototype, one productivity model, some models which are related from an ancient opinion, well documented and inside the experience of the author, and certain models which present fresh thoughts.

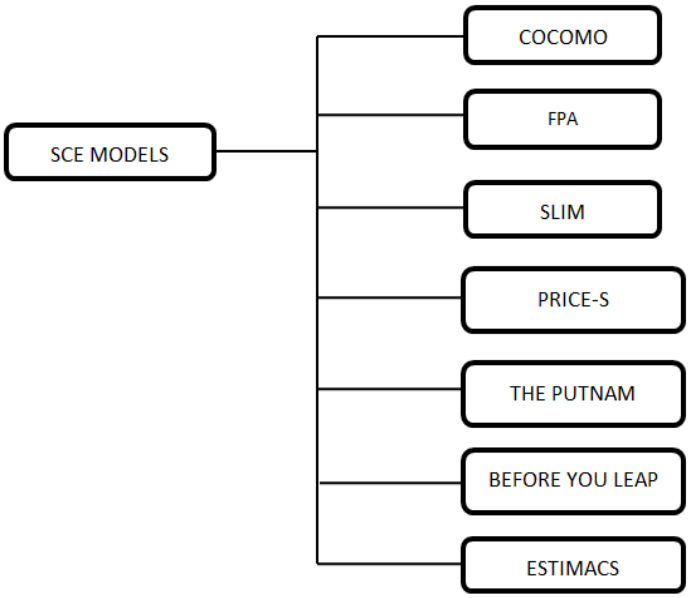

Fig 4. SCE Models

The Constructive Cost Model (COCOMO): It is the finest recognized and most obvious model presently accessible. The central attention in COCOMO is upon estimating the effect of 15 cost drivers on the growth effort. Beforehand this can be done, o software size estimation must be presented. COCOMO does not include the sizing estimation stage:

it only contributes numerous equations based on 63 finished projects at TRW. The equations signify the associations among size and effort and among effort and growth time. The equations are shown in Table 2.

TABLE II. COCOMO equations

Development mode	Man-month Development time (nominal)	Man-month Development time (nominal)
Organic	$3.2*KDSP^{1.05}$	$2.5*MM\ (nom)^{0.38}$
Semi-detached	$3.0*KDSI^{1.12}$	$2.5*MM\ (nom)^{0.35}$
Embedded	$2.8*KDSI^{1.20}$	$2.5*MM\ (nom)^{0.32}$

There are 3 modes: the organic mode which is steady development surroundings, less innovative, comparatively minor size; the embedded mode which is increasing within tight constraints, innovative, difficult, high instability of requests; and the semi-detached mode which exists among organic and embedded mode.

Function point analysis (FPA): FPA has been established by Albrecht of IBM, and made extensively obtainable via the consumer clusters Guide and Segment. Albrecht was watching for a technique to evaluate productivity in software development. For that purpose he established FPA as an alternate portion to the number of lines of code. The technique is programming language or fourth generation tool independent.. The idea of FPA is modest and is founded on the number of 'functions' the software has to achieve.

SLIM : It is one of the greatest important cost estimation models that has been in the advertise for years. The model was initially established in the 1970s by Larry Putnam of Quantitative Software Measurement, and its measured formulas and investigation were distributed in 1992. As the model is branded, the following advancements of the model organizations and mathematical formulations are not accessible in the open domain [12]. Usually, the SLIM model assumes that the recruitment outline keep an eye on a form of Rayleigh probability distribution of project staff buildup over time. The Rayleigh staffing level at time t is offered as

$$p(t) = \frac{K}{t_d^2} te^{\left(-\frac{t^2}{2t_d^2}\right)}$$

Where K is the total lifecycle effort and td the schedule to the peak of the staffing curve. The quality 2 dt KD = is considered staffing complexity of the project. The total lifespan effort is planned using the project size S, the technology factor C, and td, and is defined as

$$K = \frac{S^3}{C^3} x \frac{1}{t_d^4}$$

SLOC: The model involves the real SLOC as a unit of project size. Function points and user-defined metrics like number of units, screens, etc. can be involved, but they have to be transformed to real SLOC using a 'gear factor'. SLIM calculates fresh code and altered code, but it eliminates erased code. Obviously, the model assumes that fresh code and improved code have the same effect on the maintenance effort.

PRICE-S: The PRICE-S model stands for Programming Review of Information Costing and Evaluation—Software. It is established and sustained by RCA PRICE Schemes. An significant drawback with respect to COCOMO and FPA is that the fundamental ideas and thoughts are not openly clear and the employers are offered with the method as a black box. The consumer of PRICE directs the idea to a time-sharing computer in the USA, UK, or France and acquires back his estimations directly. Even though this drawback and the extraordinary rental price, there are various operators, particularly in America. There is, still, an significant inspiration for American corporations to consume the model.

The PUTNAM model: This SCE method was established by Putnam in 1974. He founded his model on the effort of Norden. For numerous plans at IBM, Norden planned frequency dispersals, in which he presented how many persons were distributed to the growth and repairs of a software artifact during the life-cycle. The arcs he made fixed very well with the Rayleigh arcs. His answers were just experiential. He originate no clarifications for the outline of the effort arc. On the assumptions of Norden, Putnam expressed his model.

Before You Leap (BYL): BYL is a profitable bundle based on a link-up among FPA and COCOMO. BYL initiates with a evaluation of the quantity of net function points. This quantity is then transformed into source lines of code, taking in account the language used. For Cobol, for example, one function point is equal to 105 SLOC, for LISP it is 64, etc. This guesstimate of the size in SLOC is exactly the essential idea for COCOMO and the COCOMO part of BYL, taking into account the effect on effort of the 15 COCOMO cost drivers, computes the estimates of costs and span scale.

Estimacs: Estimacs has been established by H. Rubin and Computer Acquaintances, and is obtainable as a software platform [15]. The approach contains nine units: a function point unit; a risk unit; an effort unit, etc. The greatest significant and broad unit is Effort. The consumer has to response 25 queries. These queries are partially connected to the complication of the consumer-organization and partially to the complication and size of the software to be established. The method Estimacs decodes the idea to an estimation of effort is not clear. Like several other prototypes, Estimacs is a 'closed model'.

IV. CONCLUSION AND FURURE SCOPE

Maintenance of software may be a terribly broad activity that has enhancements in capabilities, error alteration, optimization and removal of obsolete competences. Modification is predictable and therefore, mechanisms should be developed for dominant, evaluating and creating deviations. A suitable plan for maintenance action should be organized while emerging the software which is a significant feature of software conservation. This strategy should state the method in which alterations are to be prepared. The expensive to grow the software should comprise the price due to any essential alteration in the software. It means that maintenance charge is not only due to deprived design, but also due to the alteration in client beliefs and environmental requirements in which the structure has been established. In this paper various factors of Software maintenance are discussed and methods and tools for cost estimation are also discussed. The authors has defines various parameters of cost estimation and literature review related to this is also discussed. SCE models stands for software cost estimation models which helps in estimating the cost for software. In this various software estimation models such as COCOMO, Putnam, Estimacs, before you leap, price-s, SLIM are discussed in detail. In future one of the cost estimation models will be implemented.

REFERENCES

[1] Banker, R. D., Datar, S. M. and Kemerer, C. F. (1987),"Factors affecting software maintenance productivity: an exploratory study", Proceedings of 8th international conference on Information system. Pittsburgh S, pp. 160–175.

[2] Boehm B.W., Abts C. andChulani S. (2000), "Software development cost estimation approaches: A survey", Journal of Annals of Software Engineering, 10(1-4), pp. 177-205.

[3] Buchmann, Irene, Sebastian Frischbier and Dieter Putz (2011), "Towards an estimation model for software maintenance costs", Proceedings of (CSMR), 15th European Conference on Software Maintenance and Reengineering, Oldenburg, Germany, pp. 313 – 316.

[4] Choudhari, Jitender and Suman, Ugarsen (2012), "Phase wise effort estimation for software maintenance: an extended SMEEM model", Proceedings of the CUBE International Information Technology Conference, Pune, pp. 397-402.

[5] De Lucia A., Pompella E., and Stefanucci S. (2005), "Assessing effort estimation models for corrective maintenance through empirical studies", Journal of Information and Software Technology, 47(1), pp. 3–15.

[6] Dehaghani SM and Hajrahimi N. (2013),"Which factors affect software projects maintenance cost more", Journal of Acta Informatica Medica, 21(1), pp. 63-64.

[7] Grubb, Penny and Armstrong, A. (2003), "Software maintenance: concepts and practice", Singapore: World Scientific Publishing Company.

[8] Jorgensen, M. (1995), "Experience with the accuracy of software maintenance task effort prediction models", Journal of IEEE Transactions on Software Engineering, 21(8), pp. 674-681.

[9] Lehman, M.M., J.F. Ramil, P.D. Wernick, D.E. Perry and W.M. Turski (1997), "Metrics and laws of software evolution - the nineties view", IEEE International Symposium on Software Metrics (METRICS'97), Los Alamitos, CA,, pp.20-32.

[10] Lientz, B. and Swanson, E.(1980), "Software maintenance management", Boston, MA: Boston Addison-Wesley Longman.

[11] Marounek, P. (2012), "Simplified approach to effort estimation in software maintenance", Journal of systems integration, 3(3), pp. 51-63 in Jorgensen, M. (1995), "Experience with the accuracy of software maintenance task effort prediction models", Journal of IEEE Transactions on Software Engineering, 21(8), pp. 674-681.

[12] Nguyen, Vu (2010). "Improved size and effort estimation models for software maintenance", An Unpublished Ph.D. Dissertation, University of Southern California, Los Angeles, CA, viewed 20 December 2014, <http://csse.usc.edu/csse/TECHRPTS/ PhD_ Dissertations/ files/Nguyen_Dissertation.pdf.>.

[13] Ramin, Moazeni, Daneil Link and Barry Boehm (2014), "COCOMO II parameters and IDPD: bilateral relevance", Proceedings of the International Conference on Software and System Process , China,. Pp. 20-24.

[14] Sharma, T. N., A. Bhardwaj and G.R. Kherwa (2012), "Statistical analysis of various models of software cost estimation", International Journal of Engineering Research and Applications (IJERA), 2(3), pp.683-685, viewed 27 December 2014, <www.ijera.com> .

[15] Sneed, H.M. (2004), "A cost model for software maintenance and evolution", Proceedings of IEEE 20th International Conference on Software Maintenance, Chicago, USA pp. 264 – 273.

[16] Syavasya, C. V. S. R. (2013), "An approach to find maintenance costs using cost drivers of Cocomo intermediate model", International Journal of Computational Engineering Research, 3(1), pp. 154–158.