International Journal of Allied Practice, Research and Review
Website: www.ijaprr.com (ISSN 2350-1294)

Software Cost Estimation Techniques Using Soft Computing

Shiva Suryanarayana Ch.[1] and Dr. Satya Prakash Singh[2]
Computer Science & Engineering
[1]BIT, Mesra, Ranchi, Noida Campus India, India, [2]Ras Al Khaimah, Dubai Campus

Abstract: - The software cost estimation is the process of predicting the most realistic amount of effort required to develop or maintain software based on incomplete, uncertain and noisy input. Effort estimates may be used as input to project plans, iteration plans, budgets, and investment analyses, pricing processes and bidding rounds. For example, estimation of value estimation and danger examination is a real issue in programming undertaking administration. As programming advancement has turned into a fundamental speculation for some associations, precise programming expense estimation models are expected to adequately foresee, screen, control and evaluate programming improvement. Programming expense estimation is a testing and cumbersome errand. Then again, the approach utilized for the estimation of programming exertion by relationship is not ready to handle the unmitigated information in an unequivocal and exact way. The nature of an expense estimation model is less ascribed to the introductory assessment, but instead the pace at which the appraisals merges to the genuine expense of the task. COCOMO is a well known algorithmic model for expense estimation whose expense variables can be customized to the individual improvement environment, which is imperative for the precision of the expense gauges. More than one strategy for expense estimation ought to be carried out so that there is some correlation accessible for the evaluations. This is particularly imperative for extraordinary undertakings. Cost estimation must be done more diligently throughout the project life cycle so that in the future there are fewer surprises and unforeseen delays in the release of a product. In this paper, we present a soft computing framework to tackle this challenging problem. Estimating the work-effort and the schedule required to develop and/or maintain a software system is one of the most critical activities in managing software projects.

Keywords: ***Software Engineering, Software Cost Estimation, COCOMO model, Soft Computing Techniques.***

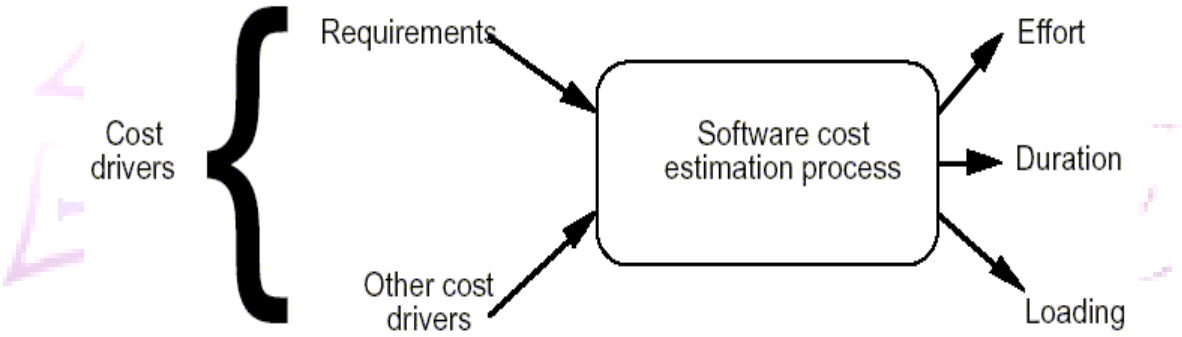
I. Introduction

Software cost estimation is process of predicting the effort required to develop a software engineering project. While the software cost estimation may be simple in concept, it is troublesome and perplexing truly. The genuine bit of cost of programming change is a direct result of human-effort and most cost estimation frameworks focus on this viewpoint and give assesses in regards to individual month It chooses the measure of effort imperative to complete an item wander in terms of its booking, securing of resources, and meeting of arrangement essentials. The effective and viable change of the programming requires exact gages. Programming researchers are giving various cost estimation frameworks for a couple of decades yet the essential issue hang on in programming building field. Early programming estimation models can't avoid being constructed in light of the backslide examination or exploratory derivations. Among those systems, COCOMO II is the most normally used

model. Today's models are concentrated around reenactment, neural framework, genetic computation, fragile handling, the soft method of reasoning showing etc. In this paper, COCOMO II model utilized the most much of the time and generally utilized hereditary calculation approach for upgrading the current coefficients that gauge the streamlined prescient exertion obliged for the advancement of programming undertaking.

1.1 Cost Estimation Process:- To comprehend the finished result or the yields of the product cost estimation process we should first comprehend what is programming expense estimation process. By definition, programming expense estimation methodology is a situated of strategies and methods that is utilized to infer the product expense gauge. There is normally a situated of inputs to the methodology and afterward the procedure utilizes these inputs to produce or ascertain a set of yields.

1.2 Classical View:-Most of the product cost estimation models sees the estimation transform just like a capacity that is registered from a situated of expense drivers. Also in most cost estimation methods the essential expense driver or the most critical expense driver is accepted to be the product necessities. As showed in figure 1, in an established perspective of programming estimation prepare, the product necessities are the essential information to the procedure furthermore structure the premise for the expense estimation. The expense assessment will then be balanced likewise to various other expense drivers to land at the last gauge. So what is expense driver? Fetched driver is anything that may or will influence the expense of the product. Taken a toll driver are things, for example, outline procedure, ability levels, hazard evaluation, staff experience, programming dialect or framework unpredictability.



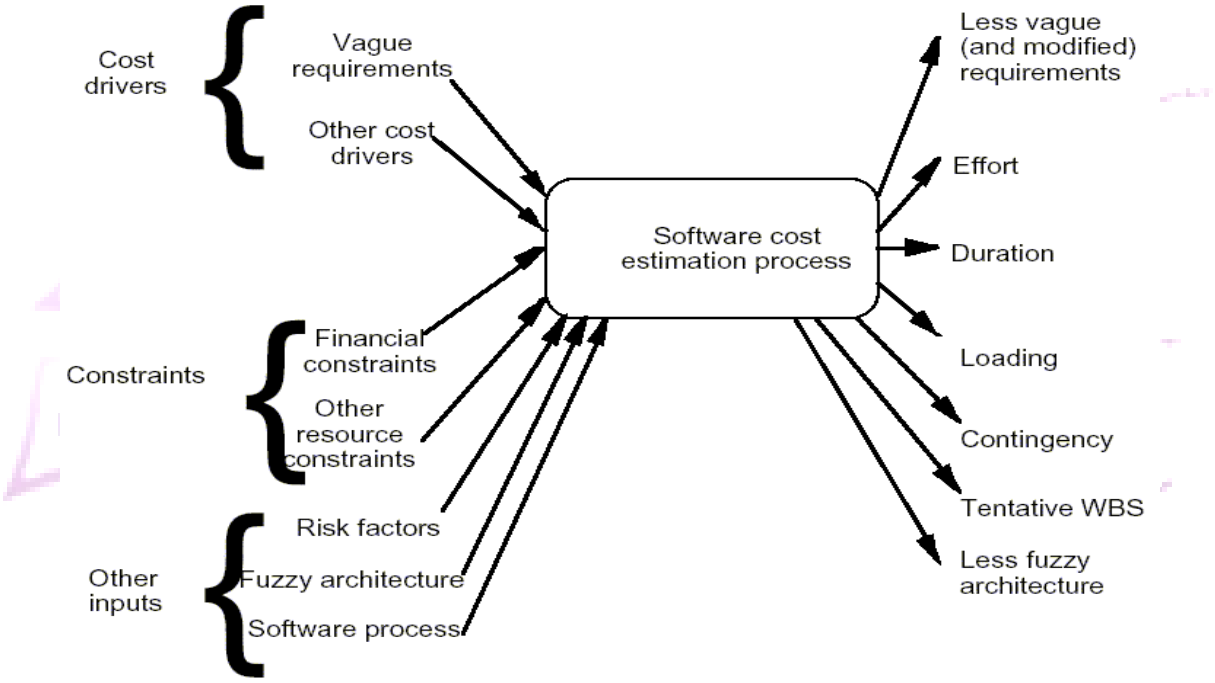
In a traditional perspective of the estimation process, it will produce three yields - endeavors, term and stacking. The accompanying is a concise depiction of the yields:

- Manpower stacking - number of work forces (which likewise incorporates administration faculty) which are distributed to the venture as a capacity of time.
- Project term - time that is expected to finish the task.
- Effort - measure of exertion needed to finish the undertaking and is generally measured in units as man-months (MM) or individual months (PM).

The yields (stacking, term and exertion) are generally registered as settled number with or without resilience in the traditional perspective. However actually, the expense estimation procedure is more unpredictable than what is indicated in figure 1. Large portions of the information that are inputs to the methodology are altered or refined amid the product cost estimation process.

1.3 Actual View: In the actual cost estimation process there are other inputs and constraints that needed to be considered besides the cost drivers. One of the primary constraints of the software cost estimate is the financial constraint, which are the amount of the money that can be budgeted or allocated to the project. There are other constraints such as manpower constraints, and date constraints. Other input such as architecture, which defines the components that made up the system and the interrelationships between these components. Some company will have certain software process or an existing architecture in place; hence for these companies the software cost estimation must base their estimates on these criteria.

There are only very few cases where the software requirements stay fixed. Hence, how do we deal with software requirement changes, ambiguities or inconsistencies? During the estimation process, an experienced estimator will detect the ambiguities and inconsistency in the requirements. As part of the estimation process, the estimator will try to solve all these ambiguities by modifying the requirements. If the ambiguities or inconsistent requirements stay unsolved, this will correspondingly affect the estimation accuracy.



1.4 Cost Estimation Accuracy: The cost estimation accuracy helps to determine how well or how accurate our estimation is when using a particular model or technique. We can assess the performance of the software estimation technique by:

- Absolute Error (Epred - Eact)
- Percentage or Relative Error (Epred - Eact) / Eact
- Mean Magnitude of Relative Error

Each of the error calculation techniques has advantages and disadvantages. For example, absolute error fails to measure the size of the project, and mean magnitude of relative error will mask any systematic bias

Method of Cost Estimation: There is a lot of software cost estimation methods or techniques in the software industry. Here are a few techniques that will be discussed in this Paper:

- Algorithmic model
- Expert Judgment (Expertise Based)

- Top - Down
- Bottom - Up
- Estimation by Analogy
- Price to Win Estimation

II. Algorithmic Model

These product cost estimation procedures utilize the numerical mathematical statements to perform the product estimation. The numerical correlations are concentrated around chronicled data or theory. SLOC (source line of code), limit centers, and other cost drivers are the inputs. For most algorithmic model, arrangement to the specific programming environment can be performed to improve the estimation. Delineations of the parametric models are COCOMO (Constructive Cost Model), COCOMO II, Putnam's item life-cycle model (SLIM).

As per paper, COCOMO II post structural engineering system ascertains the product advancement exertion (in individual months) by utilizing the accompanying comparison:

Effort = A × (SIZE) E × Πi EMi. …… (1)

Where, A- multiplicative constant with value 2.94 that scales the effort according to specific project conditions. Size - Estimated size of a project in Kilo Source Lines of Code or Unadjusted Function Points. E - An exponential factor that accounts for the relative economies or diseconomies of scale encountered as a software project increases its size

EMi - Effort Multipliers.

The coefficient E is determined by weighing the predefined scale factors (SFi) and summing them using following equation:

E = 0.91 + 0.01 ∑i SFi ...…….. (2)

The development time (TDEV) is derived from the effort according to the following equation: TDEV = C × (Effort) F ...… ……(3)

Latest calibration of the method shows that the multiplier C is equal to 3.67 and the coefficient F is determined is a similar way as the scale exponent by using following equation: F = 0.28 + 0.002 ∑i SFi ...… ……(4)

According to paper [6], when all the factors and multipliers are taken with their nominal values, then the equations for effort and schedule are given as follows:

Effort = 2.94 × (Size) 1.1 …..... (5)

Duration: TDEV = 3.67 × (Effort) 3.18 ...… (6)

COCOMO II is clear and effective calibration process by combining Delphi technique with algorithmic cost estimation techniques. It is tool supportive and objective. This model is repeatable, versatile. But its limitation is that most of extensions are still experimental and not fully calibrated till now.

Advantages:

- Generate repeatable estimations
- Easy to adjust information
- Easy to refine and modify equations
- Objectively aligned to experience

Disadvantages:

- Unable to manage remarkable conditions
- Some experience and components cannot be measured
- Sometimes calculations may be restrictive

2.1 Expert Judgment

This method catches the experience and the information of the estimator who gives the appraisal focused around their experience from a comparative task to which they have partaken. Illustrations are the Delphi, Wideband Delphi and Work Breakdown Structure (WBS).

Preferences:

- Useful without measured, experimental information.
- Can figure in contrasts between past venture encounters and necessities of the proposed undertaking
- Can calculate in effects brought on by new innovations, applications and dialects.

Disadvantages:

- Estimate is just as great master's supposition
- Hard to archive the components utilized by the masters

2.2 Top-Down

This strategy is additionally called Macro Model, which use the worldwide perspective of the item and afterward parceled into different low level segments. Illustration of this procedure is the Putnam model.

Points of interest:

- requires negligible undertaking subtle element
- usually quicker and simpler to execute
- focus on framework level exercises

Drawbacks:

- tend to ignore low level part

III. Literature Review

Numerous programming expense estimation models have been produced in the course of the most recent decades. A late study by Jorgensen gives a point by point survey of diverse studies on the product improvement exertion. Numerous analysts have connected the neural systems approach to gauge programming advancement exertion. A wide range of models of neural systems have been proposed. They may be assembled in two noteworthy classifications. Initial one is nourishing forward systems where no circles in the system way happen. Another is criticism arranges that have recursive circles. Understanding the affliction in applying neural systems, Nasser Tadayon has proposed a dynamic neural system that will at first use COCOMO II Model. COCOMO, be that as it may, has some limits. It can't adequately bargain with uncertain and indeterminate data, and alignment of COCOMO is a standout amongst the most essential errands that need to be carried out so as to get precise estimations. Thus, there is dependably scope for creating exertion estimation models with better prescient precision.

- **Some example of estimation methods inside each class:-**

Estimation approach	Category	Examples of support of implementation of estimation approach
Analogy-based estimation	Formal estimation model	ANGEL, Weighted Micro Function Points
WBS-based (bottom up) estimation	Expert estimation	Project management software, company specific activity templates
Parametric models	Formal estimation model	COCOMO, SLIM, SEER-SEM, True Planning for Software
Size-based estimation models	Formal estimation model	Function Point Analysis, Use Case Analysis, SSU , Story points-based estimation in Agile software development
Group estimation	Expert estimation	Planning poker, Wideband Delphi
Mechanical combination	Combination-based estimation	Average of an analogy-based and a Work breakdown structure-based effort estimate
Judgmental combination	Combination-based estimation	Expert judgment based on estimates from a parametric model and group estimation

IV. COCOMO

COCOMO is no doubt the most popular method for doing software cost estimation. The estimations are relatively easy to do by hand. There also are tools available which allow you to calculate more complex estimation. Calibration of COCOMO is one of the most important things that needs to be done in order to get accurate estimations. Even though COCOMO may be the most popular estimation method it is recommended that you always use another method of estimation to verify your results. The other method should differ significantly from COCOMO. This way your project is examined from more then one angle and something that you may have overlooked when using COCOMO is not overlooked again.

COCOMO stands for Constructive Cost Model, it is a software cost estimation model that was first published in 1981 by Barry Bohem (2001) It is an algorithmic approach to estimating the cost of a software project. By using COCOMO you can calculate the amount of effort and the time schedule for projects. From these calculations you can then find out how much staffing is required to complete a project on time. COCOMO's main metric used for calculating these values is lines of code (denoted KLOC for COCOMO II, or KDSI for COCOMO 81 and measured in thousands), function points (FP), or object points (OP).

COCOMO also lets you check out 'what if' scenarios where by adjusting certain factors in COCOMO you can see how a projects time and effort estimates change as well(Bohem2001). There have been a few different versions of COCOMO; the two that are discussed in this report are COCOMO 81 and COCOMO II.

V. COCOMO 81

COCOMO 81 was the first version of COCOMO. It was modeled around software practices of the 1980's. It has been found that on average it is able to produce estimates that are within 20% of the actual values 68% of the time. COCOMO 81 has three different models that can be used throughout a projects life cycle (Bohem,2001)

Basic Model – this model would be applied early in a projects development. It will provide a rough estimate early on that should be refined later on with one of the other models.

- Intermediate Model – this model would be used after you have more detailed requirements for a project.
- Advanced Model – when your design for a project is complete you can apply this model to further refine your estimate.

Within each of these models there are also three different modes. The mode you choose will depend on your work environment, and the size and constraints of the project itself. The modes are:

- Organic – this mode is used for "relativity small software teams developing software in a highly familiar, in-house environment".
- Embedded – operating within tight constraints where the product is strongly tied to a "complex of hardware, software, regulations and operational procedures".
- Semi-detached – an intermediate stage somewhere in between organic and embedded. Projects are usually of moderate size of up to 300,000 lines of code.

Equations Used

There are two main equations that are used to calculated effort and schedule time (measured in months). They are:

Equation 1 PM = a(KDSI)b * EAF

Equation 2 TDEV = c(PM)d

Where:

- PM is effort in person-months
- EAF is the effort adjustment factor
- TDEV is the schedule time
- KDSI is the number of lines of code (in thousands)
- a, b, c, and d are all constants based on the mode you are using (refer to Table 1)
-

Table 1 – List of Constants Based on Mode

Model	A	B	C	D
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

5.1 Cost Drivers

The EAF is used to tailor your estimate based on conditions of the development environment. For the basic model it is not used and just set to 1. For the intermediate model there are 15 different cost drivers that can be used to calculate your EAF. They are grouped into 4 different categories; product attributes, computer attributes, personal attributes, and project attributes (see Table 2). Each cost driver is rated on a scale Very Low to Extra High depending on how that cost driver will affect your development. These ratings are based on a statistical analysis of historical data collected from 83 past projects.

To calculate the EAF from the cost drivers you simply choose values for each cost driver and multiply them all together. The resulting number is your EAF.

Table 2. List of 15 costs drivers and their ratings for COCOMO 81

Category	Cost Driver	Very Low	Low	Nominal	High	Very High	Extra High
Product Attributes	RELY Required Software Reliability	0.75	0.88	1.00	1.15	1.40	-
	DATA Database Size	-	0.94	1.00	1.08	1.16	-
	CPLX Product Complexity	0.70	0.85	1.00	1.15	1.30	1.65
Computer Attributes	TIME Execution Time Constraint	-	-	1.00	1.11	1.30	1.66
	STOR Main Storage Constraint	-	-	1.00	1.06	1.21	1.56
	VIRT Virtual Machine Volatility	-	0.87	1.00	1.15	1.30	-
	TURN Computer Turnaround Time	-	0.87	1.00	1.07	1.15	-
Personnel Attributes	ACAP Analyst Capability	1.46	1.19	1.00	0.96	0.71	-
	AEXP Applications Experience	1.29	1.13	1.00	0.91	0.82	-
	PCAP Programmer Capability	1.42	1.17	1.00	0.86	0.70	-
	VEXP Virtual Machine Experience	1.21	1.10	1.00	0.90	-	-
	LEXP Language Experience	1.14	1.07	1.00	0.95	-	-
Project Attributes	MODP Modern Programming Practices	1.24	1.10	1.00	0.91	0.82	-
	TOOL Use of Software Tools	1.24	1.10	1.00	0.91	0.83	-
	SCED Required Development Schedule	1.23	1.08	1.00	1.04	1.10	-

The advanced model of COCOMO 81 goes one step further then the intermediate modem in that it uses costs drivers that are rated differently depending on the current phase that a project is in. One of the problems with using a model like COCOMO 81 today is that it does not match the develop environment of the late 1990's and 2000's. It was created in a time when batch were the norm, programs were run on mainframes and compile times were measured in hours not seconds. It is outdated for use in today's development environment (rapid application development, 4th generation languages etc) so in 1997 COCOMO II was published and was suppose to solve most of these problems

5.2 COCOMO II

COCOMO II was published in 1997 and is an updated model that addresses the problems with COCOMO 81. The main objectives of COCOMO II were set out when it was first published. They are:

- To develop a software cost and schedule estimation model tuned to the life cycle practices of the 1990's and 2000's.
- To develop software cost database and tool support capabilities for continuous model improvement.
- To provide a quantitative analytic framework, and set of tools and techniques for evaluating the effects of software technology improvements on software life cycle costs and schedules.

For the most part estimates are obtained in pretty much the same way as COCOMO 81. The main changes have been in the number and type of cost drivers and the calculation of equation variables rather then the use of constants (for a detailed look at the specific differences between COCOMO 81 and COCOMO II see). The equations still use lines of code as their main metric, you can however also using

function points and object points to do estimates. The line of code metric used is now the LOC. There are standards set out by SEI for proper counting of lines, things like if/then/else statements would be counted as one line (there are automated tools that will do the counting for you when you want to collect data from your own code).

COCOMO II again has three models, but they are different from the ones for COCOMO 81. They are:

- Application Composition Model – this would be used for projects built using rapid application development tools. Normally you would use object points for size estimates. It "involves prototyping efforts to resolve potential high-risk issues such as user interfaces, software/system interaction, performance, or technology maturity.".
- Early Design Model – This model can provide you with estimates early in a projects design before the entire architecture has been decided on. Normally you would use function points as a size estimate. It "involves exploration of alternative software/system architectures and concepts of operation. At this stage, not enough is generally known to support fine-grain cost estimation.".
- Post-Architecture Model – The most detailed on the three, used after the overall architecture for the project has been designed. You could use function points or LOC's for size estimates. It "involves the actual development and maintenance of a software product"

5.2.1 Cost Drivers

In COCOMO II there are 17 cost drivers that are used in the Post-Architecture model. They are used in the same way as in COCOMO 81 to calculate the EAF. The cost drivers are not the same ones as in COCOMO 81; they are better suited for the software development environment on the 1990's and 2000's. They are grouped together as shown in table 3. We will not go into specific details on all of the cost drivers here as that information can be found in the paper "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0". The cost drivers for COCOMO II are again rated on a scale from Very Low to Extra High in the same was as in COCOMO 81.

Example:- Assume that the size of an organic type software product has been estimated to be 32,000 lines of source code. Assume that the average salary of software engineers be Rs. 15,000/- per month. Determine the effort required to develop the software product and the nominal development time. From the basic COCOMO estimation formula for organic software:

Effort = 2.4 x (32)1.05 = 91 PM

Nominal development time = 2.5 x (91)0.38 = 14 months

Cost required to develop the product = 14 x 15,000 = Rs. 210,000/-

VI. Conclusion

The paper suggests soft computing approach for estimating of software project development cost and time. Most important issue in software project management is accurate and reliable estimation of software development effort and cost. This is more essential particularly in the early period of programming improvement so that the chief may submit his assets for on time conveyance of the product. Programming advancement is famous for going after some time and plan. This issue is because of the way that product improvement is a complex methodology due to the quantity of components included, including the human element, and the many-sided quality of the item that is created. Besides, the business is exceedingly aggressive. COCOMO is a prominent exact estimation display that has been joined into a few devices. Any model ought to be adjusted to the improvement environment on the grounds that all advancement situations are distinctive. In the end, an accurate estimate cannot be guaranteed and so using more than one method of estimation is recommended for verification of an estimate.

VII. References

1. Verma H.k and Sharma V. (2010). Taking care of Imprecision's in Inputs utilizing Fuzzy Logic to anticipate exertion in Software Development. Processes of IEEE International Advance Computing Conference, ISBN: 978-1-4244-4790-9, pp: 436-442.

2. Bhandhari S., Parveen Kakkar(2013) "Delicate registering based method for precise exertion estimation: A study" An International Journal of Engineering Sciences, Issue December 2013, Vol. 9

3. Huang, X., Ho, D., Ren, J. also Capretz, L., (2007), "Enhancing the COCOMO Model with a Neuro Fuzzy Approach," Computer Journal of Applied Soft Computing Journal, Vol. 7, No. 3, pp. 29-40 .

4. Zheng, Y., Wang, B., Zheng, Y., Shi, L., (2009), "Estimation of programming undertakings exertion focused around capacity point", In Proceedings of fourth International Conference on Computer Science & Education.

5. Xishi Huang, Danny Ho Jing Ren.(2006)"a delicate figuring structure for programming exertion estimation" Soft Comput (2006) 10: 170–177.

6. Abeer Hamdy(2012)"fuzzy Logic for Enhancing the Sensitivity of COCOMO Cost Model" VOL. 3, NO. 9, SEP 2012.

7. Boehm, B.w., Valerdi, R., Lane, J., and Brown, A.w.: 'COCOMO Suite Methodology and Evolution', The Journal of Defense Software Engineering, 2005, pp. 20-25

8. S. A. Abbas, X. F. Liao, S. U. Lar and R. A. Naseem, "Programming Models, Extensions and Independent Modelsin Cocomo Suit: A Review," Journal of Emerging Trends in Computing and Information Sciences, Vol. 3, No. 5, May 2012.

9. G.R. Finnie and G. E. Wittig, "A examination of programming exertion estimation strategies: utilizing capacity focuses with neural systems, case based thinking and relapse models," J. Frameworks Software, vol. 39, pp. 281-289, 1997.

10. K. V. Kumar, V. Ravi M. Carr, and N. R. Kiran, "Programming improvement cost estimation utilizing wavelet neural systems," Journal of framework and programming, vol. 81, pp. 1853-1867, 2008.

11. B. Samson, D. Ellison, and P. Dugard , "Programming Cost Estimation Using and Albus Perceptron (CMAC), " Information and Software Technology, vol. 39, pp. 55-60, 1997.

12. K. Subba Rao,s Reddy(2013)" Software Cost Estimation in Multilayer Feed forward Network utilizing Random Holdback Method" Volume 3, Issue 10, Pp.1309-26.

13. Magne J and Martin S, ― A Systematic Review of Software Development Cost Estimation Studies , IEEE Transactions On Software Engineering, Vol. 33, No. 1, pp. 33-53, January 2007.

14. K. Vinaykumar, V. Ravi, M. Carr and N. Rajkiran, ―software cost estimation utilizing wavelet neural networks,journal of Systems and Software, pp. 1853-1867, 2008.

15. M.v. Deshpande and S.g. Bhirud, ―analysis of Combining Software Estimation Techniques, International Journal of Computer Applications (0975 – 8887) Volume 5 – No.3, 2010 .

16. Amanjot Singh Klair and Raminderpreetkaur Software Effort Estimation utilizing SVM and knn International Conference on Computer Graphics, Simulation and Modeling (Icgsm'2012) July 28-29, 2012.

17. B. Boehm, Software Cost Estimation with COCOMO II, Prentice Hall PTR, Upper Saddle River, 2000, p. 17

18. Yan-Fu Li, Min Xie, Thong-Ngee Goh,(2010): Adaptive edge relapse framework for programming expense assessing on multi-collinear datasets , The Journal of Systems and Software, Vol 83, pg. 2332-2343.

19. Prasad Reddy P.v.g.d, Sudha K.r., Rama Sree P & Ramesh S.n.s.v.s.c, (May 2010):software Effort Estimation utilizing Radial Basis and Generalized Regression Neural Networks, Journal of Computing, Vol 2, Issue 5.

20. Parvinder S. Sandhu, Manisha Prashar, Pourush Bassi, and Atul Bisht,(2009): A Model for Estimation of Efforts in Development of Software Systems , World Academy of Science, Engineering and Technology, No.56.

21. Zia.z, Rashid.a, .Zaman.k.uz, (2009): Software cost estimation for segment based fourth era dialect, Vol.5, Iss.1, pp.103-110, IET Software. A Fuzzy Logic Based Software Cost Estimation Model

22. Ziauddin, Shahid Kamal, Shafiullah khan and Jamal Abdul Nasir(2013)" A Fuzzy Logic Based Software Cost Estimation Model" International Journal of Software Engineering and Its Applications Vol. 7, No. 2, March, 2013.

23. Abts C, Clark B, Devnani Chulani S, Horowitz E Madachy R Reifer D, Selby R., and Steece B (1998) COCOMO II Model Definition Manual. Focus for Software Engineering, University of Southern California.

24. Bogdan Stępień, "Software Development Cost Estimation Methods and Research Trends" Computer Science, Vol. 5, 2003, pp. 68-82

25. Prasad Reddy.p.v.g.d & Ch.v.m.k.hari (2011)" Software Effort Estimation Using Particle Swarm Optimization With Inertia Weight"international Journal of Software Engineering (IJSE), Volume (2) : Issue (4) : 2011 , 87-96
.
26. Shaveta Gupta, Jimmy Singla(2013)" Various Analysis and Design Techniques for Software Engineering Model utilizing Soft Computing" International Journal of Computer Science and Communication Engineering, Special issue on "Late Advances in Engineering & Technology" NCRAET-2013.