VOL. 3, NO. 4, April 2012 ISSN 2079-8407
Journal of Emerging Trends in Computing and Information Sciences
©2009-2012 CIS Journal. All rights reserved.

http://www.cisjournal.org

Cost Estimation: A Survey of Well-known Historic Cost Estimation Techniques

[1] Syed Ali Abbas, [2] Xiaofeng Liao, [3] Aqeel Ur Rehman, [4] Afshan Azam, [5] M.I. Abdullah

[1] College of Computer Science, Chongqing University, PR China
[2] Professor, College of Computer Science, Chongqing University, PR China
[3] College of Computer Science, Chongqing University, PR China
[4] College of Economy and Bussiness Administration, Chongqing University PR China

ABSTRACT

Number of contributors has made their efforts to produce different modeling techniques in last four decades. This paper is about the comprehensive descriptive exploration of the models that were presented in the early stages of the software estimation field and covers most of the famous available and practiced parametric models and few non-parametric techniques. All widespread models discussed at one place will give our readers a prospect to comprehend the pros and cons, similarities and the differences among these models

Key Words: *Software cost Estimation models; Effort Prediction; analogy; software Metrics; line of code.*

1. INTRODUCTION

Software Cost Estimation is an important, essential and a difficult task since the foundation of computer was laid in 1940's. As the variation occurred in software size from small to medium or large based, the need for precision or correctness in software cost estimation with understanding has also grown [1][2].

Understanding Software cost is important because the overall impact of the costs on any development project is large. The rate of the development of new software is much less than the abilities to develop new software [23]. While keeping resources and abilities in mind, the development rate can be accelerated, but this may affect the quality attributes of the developing software project. However, by means of modern cost effective techniques, the reduction in development costs and improved software quality can be achieved [2].

We leave the issue of reducing cost and enhancing the quality to some other paper, here we come back to the actual topic that is "cost estimation" which refers to the process of estimating the cost and time required for the development of software [13].

The author in [13] suggested estimation as the predictions of the likely amount of effort, time and staffing levels required to build the software system.

In other words it is depicted as the process of forecasting or estimating the effort required to develop a software system [4]. Software cost estimation is vital in the perspective of software engineering as a method to obtain effort, cost and time of a project, as a result helping better decision making about the feasibility or viability of the project [15]. Estimating cost with accuracy allows the project management to effectively organize the project tasks and make considerable economical and strategic planning [13]. However, cost estimation is not a simple process as it appears to be. Several known or unknown factors influence the process of estimation. For example imprecise and drifting requirements, newness (complete project or technology or both), trying to match the estimates with available time and budgets, impractical or heavy change in plan during the execution of the project, Software type, Programming Languages, teams capability and the stage during the development when you make the estimates. Barry Boehm in [5] has emphasized that the uncertainty is greatest at the beginning of a project and decreases as the project progresses. As the project tasks are completed by the team, the rate of change of a system's estimated final size and the uncertainty about a system's estimated final size should approach zero [6]. Another worthy factor is software size that impacts the process of cost estimation. The precision of size estimation directly impacts the accuracy of cost estimation [55].

In the competitive environment of software industry, the victorious organization will be that one, which has the capability to develop and deliver the software product to the customers or end users with in the promised period of time while staying in financial budgetary boundaries. Hence, proper estimates are the drivers which may steer to achieve these milestones. In other words it may be said that it is quiet necessary to understand and control the cost by proper estimation for the proper management, enhanced quality and better understanding of the software project [2].

Estimation, as being the sub phase of software engineering, needs to be dealt in some predefined, preplanned and in well mannered way. So, rather making wild guesses, Estimation should be made by practicing some good pre-defined method either theoretical based on judgments or mathematical approach supported with proven formulas and quantitative software metrics to make estimation measurements easy and trustworthy up to some extent.

In past, contributors have made their efforts to propose different modeling techniques, for example, Farr and Zagorski model (1965), Aron model (1969), Walston and Felix model (1970), Wolverton model (1974), Kustanowitz model (1977),Putnam SLIM(1977), GRC model (1974), Doty model (1977), Jensen model (1979), Bailey and Basili Model(1981), and so on.

This paper is the brief introduction of software metrics, a comprehensive descriptive exploration of the models that were presented in the early stages of the software estimation field and covers most of the famous available and practiced. Some theoretical approaches based on intuitions of expert personals and the evolutions of Fuzzy logic and neural networks in cost estimation are partially discussed in the paper. All widespread models discussed at one place will give our readers an option to comprehend the pros and cons, similarities and the differences among these models. The rest of the paper is distributed in following sections.

Section 2 includes the understanding of the software metrics. Section 3 & 4 include the algorithmic models and the non algorithmic modeling techniques respectively. In section 5 we have presented future work and finally in section 6 we concluded our discussion followed by Appendixes and references.

2. SOFTWARE METRICS

Many researchers have been made to identify different methods or techniques which may provide a good way to deal with the size estimation. Results of these researches provided a number of techniques, professionally speaking called software Metrics. Now days, Quantitative measure is necessary in every field of science and in computer sciences size metrics is a way to conduct quantifiable measurements, which if used properly makes cost estimation process much easier and trustworthy. A number of software metrics are proposed but following are the few which are commonly known and being practiced by different organizations.

2.1.1 Line of Code

According to the authors in [62] and further elaborated by [55], “LOC is the number of lines of the *delivered source code* of the software, excluding comments and blank lines”. The lines of code measures are the most traditional measures used to quantify software complexity and for estimating software development effort. LOC is the oldest of size estimation techniques, however productivity through LOC is not considered to be a good source to generate estimates. The reason for considering LOC old-fashioned is its language dependency. It is obvious that projects developed by the use of different programming languages even to provide the same function need different varied time and effort because of the differences between High level and low level languages [57]. Another problem with LOC is any agreed classification for a line of code. In the presence of a number of languages, there could be a lot of variations in the line counting method. Author in [55] has discussed 11 significant variations in line counting method.

Even in the presence of these drawbacks, LOC was practiced by different organizations and a number of software estimation models proposed in early days (CoCoMo, Putnam, Walston & Felix…) were the function of line of code [83].

2.1.2 Function Point Analysis

LOC is programming language dependent and obviously the productivity of LOC is effected significantly with the development of the language [57][75][69]. In such circumstances the need of Function point is emerged which measures the size of a system from its functionality and usability [10]. Pressman in [79] suggests that “Function points are derived using an empirical relationship based on countable measures of software’s information domain and assessment of software complexity”. Function points were introduced by Albrecht [76] with the objective “…to count the numbers of external user inputs, inquiries, outputs and master files to be delivered … these factors are manifestations of any application”. These factors include all functions in any given application and all these factors are counted separately and weighted by numbers reflecting the relative value of the function and these weighted counts are then summed up to yield Function point or unadjusted function points (term used by Albrecht) [77][75]. 2

The expression used for function points calculation is:

Function points delivered= Unadjusted function points * Complexity adjustment factor

The complexity adjustment factor is equal to (0.65+ 0.01(N)) where N is the sum of degree of influence of 14 factors discussed in [79] and each factor takes the value ranges from 0(very low/none) to 5(very high). This value of N is used to develop adjustment factor ranging from 0.65 to 1.35 hence providing the adjustment of +/- 35 % [75].

Function points can be computed early in the development cycle hence raising its worthiness as compared to other sizing metrics. The function point metric is perhaps one of the most successful and fast growing size metric for measuring size as a substitute to LOC [80].

2.1.3 Software Science (Halstead's Equations)

In previous sections, it is mentioned that writing code can be easy or difficult if we talk in the perspective of low level and high level language. So LOC estimation does not remain effective, and solution to eradicate this ineffectiveness is to give more weight to lines that are more complex. This thought is developed by Maurice Halstead in his metric called Software Science [82]**.** In order to estimate the code length, volume, complexity and effort, software science suggests the use of operators and operands [80].

According to Halstead Code length is used to measure the source code program length and Volume corresponds to the amount of required storage space and both are defined in [55] as:

Program length(N) = N1 + N2
Volume (V) = N log (n1+n2)

Where

n1 = number of distinct operators in a program
n2 = number of distinct operands in a program
N1 = number of occurrences of operators in a program
N2 = number of occurrences of operands in a program

Following equations are used for computing estimation [80].

N = Observed Program Length = N1 + N 2
N = Estimated Program Length
= n1 (log2 (n,))+ n2 (log2 (n2))
n = Program Vocabulary = n1 + n2
V = Program Volume = N(log 2 (n))
D = Program Difficulty = (n,/2)/(N 2 /n.)
L = Program Level = *1/D*
E = Effort = *V/1*

This work by Halstead remained prominent in the industry for small period of time. Halstead equations can not be proven successfully a better approach as compared to LOC [80]. The authors in [78] have investigated software science in detail and questioned the effectiveness of software science. They stated "…The failure to state the relationships statistically, which would permit description of the dispersion, seems to be a serious weakness both theoretically…and practically. … The standard of experimental design is frequently very poor"**.** Due to the disagreements [78], the software science has lost its support in recent years.

Few other recently proposed metrics are Weighted Micro Function Points (WMFP) uses a parser to understand the source code breaking it down into micro functions and derive several code complexity and volume metrics, which are then dynamically interpolated into a final effort score [66]. Feature Point [23] and full function points [19] are other two extensions of Function point. Object-oriented (OO) metrics that is based on the number and complexity of the object like screen, reports components etc [55] is evolving gradually.

Up to now, we have seen that cost estimation is a way to predict the resources required for any software project, influenced by a number of factors. However, this influence can be mitigated by utilizing development tools, by following an appropriate process, establishment and management of a good measurement technique. Avoiding the redundant activities is another good practice which may affect the estimation process. Researches have shown that lessening the work by eradicating un-necessary activities can raise the output to 80% [51]. At this point it may be said that a good measurement or estimate does not guarantee the successful completion of the project but it at least provides a way to manage the resources and control the costs. It is also worthy to mention that one metric probably alone is not enough to determine any information about an application under development. Several metrics may be used in process cycle to increase insight into improvements during a software development.

Review

Software cost estimation is important for project activities like planning, risk analysis, budgeting etc and if the project developing organization is lacking in the exercise of some reasonable estimation technique, then it means their projects are at risk [1]. Several models have been proposed in last thirty years; some of them survived because of their effectiveness, other was limited to organizational used, hence not practiced a lot. This section is an effort to review most of the famous estimation models which may provide an insight to readers to compare their pros and cons or to make judgment that in any particular scenario which model may be adopted to conduct estimation.

The models, which we will review in upcoming sections, can be classified in many ways and from past literature different approaches, categories and techniques for estimation modeling is identified. Authors in [74] categorized the estimation models into *Sparse Data Methods* which can be applied without depending upon historical data and *Many Data Methods* that are either function or arbitrary function models. Basili [41] has categorized these models into following four parts: (i) Static single variable model (ii) Static multivariable models (iii) Dynamic multivariable models and (iv)Theoretical models. Another classification is made by Kitchenham [36] is popularly known as *Constraint models* which are the set of the models specify the relationship between different cost parameters (e.g., cocomo, Putnam, Jensen…). Boehm and others in [73] define six major categories for estimation models. (i) Model based, (ii) expertise based, (iii) case based, (iv)Dynamic based, (v) regression based and (vi) composite based. In [55] authors have distributed models into two divisions. *Algorithmic* (also known as parametric models) generate a cost estimate by the means of one or more mathematical algorithms using variables considered to be the major cost drivers. These models estimate effort or cost based mainly on the Hardware/Software size, and other productivity factors known as cost driver attributes [65] [72]. At another place [71] suggested that algorithmic models are dependant upon the statistical analysis of historical data. On the other hand Non-Algorithmic category is an approach that is soft computing based when fuzzy logic, Neural networks and genetic algorithms are involved [68]. Techniques like Delphi, analogy and expert judgment are the non algorithmic techniques where human experience is used to make estimates by comparisons of previous work or educated guessing [70] [73].

From the wide-ranging review of the literature, we have categorized the models of our discussion in the two broad categories, Non Algorithmic and Algorithmic which is further broken down in Discrete models, Power Function models, Linear/non linear models, Multiplicative models and others. Figure I include an overview of models in their respective categories [60][73][67][55][72][74].

The following section is a brief overview of Algorithmic models given in Figure I.

3. ALGORITHMIC MODELS

3.1 Linear / Non Linear Models

Linear models rely upon derived equations from the test data; however during software development a linear model works very well because of non-linear interactions [5]. Non linear models usually uses linear estimation iteratively applied to linear approximations until coefficients converge.
The models in these two categories are

3.1.1 Bailey & Basili

This model is based on the early work of Walston and Felix (3.3.1) and was proposed to be used to express the size of the project with some measures like Line of code, executable statement, machine instructions, number of modules and a base line equation to relate this size to effort [3]. This model has also suggested an approach that is related with the project's deviation; however we will focus only upon the effort estimation proposed by Bailey & Basili.

It was discovered by authors that rather considering only the total lines or only new lines to determine the size of a project; a better way is to use an algorithm to combine these sizes into one measure. By keeping this in mind they suggested that a baseline relationship of lower standard error can be derived by computing the effective size in lines to be equal to total number of new lines plus 20% of old lines used in the project [3]. They called this new size measure developed lines *DL* and they adopted same approach to measure developed modules. The equation provided by authors with 1.25 standard error estimate, further discussed at [72], is given as:

$$\text{Effort} = 0.73 * DL^{1.16} + 3.5$$

3.1.2 Farr & Zagorski

Probably this is the earliest known model proposed around 1965 at ICC symposium on economics of automatic data processing [11]. This linear model proposed productivity factors i-e delivered statements, size of data base etc that are determined by regression analysis of three linear equations resulting into the effort required in man months to complete the system. [11][58][4]. Coefficients and Cost Drivers in the Farr-Zagorski Study with their values are given at table VI (Appendix B). The equation that can be used to estimate effort in MM and cost is given as [58].

$$MM = \sum_{x,j=0}^{6} x_i\, j_i$$

$$\text{Total Cost} = MM\ (\text{Labor rate}\ /12)$$

Authors in [45] emphasized that Linear models are not proven to be satisfactory for effort estimation however, the results computed by Mohanty[58] have proved that Farr and Zagorski model's results are better than Naval Air Development Center model and some others.

3.1.3 Nelson Model

Nelson model is a linear model that was developed by Nelson at System Development Corporation (SDC) in 1966 which was refined by him in 1970. During the development of this model, Nelson studied 169 databases of different projects. These projects were selected from different field's i.e Business, Scientific, Computer software and other. Initially Nelson identified 104 attributes and later only most significant 14 cost drivers (Table V, appendix B) were used in estimation process [49]. In his work, Nelson proposed equations for estimating manpower, computer usage, and months elapsed. These equations were proposed only for the Computer Program Design, Code and Test activity. For the total sample of 169 projects, Nelson has computed Total Man months, Total computer Hours and Total Elapsed time as follows:

Total Man month = $-33.63 + 9.15\ X_3 + 10.73\ X_8 + .51\ X_{26} + .46\ X_{30} + .40\ X_{41} + 7.28\ X_{42} - 21.45\ X_{48.1} + 13.53\ X_{48.5} + 12.35\ X_{51} + 58.82\ X_{53} + 30.61\ X_{56} + 29.55\ X_{72} + .54\ X_{75} - 25.20 X_{76}$

Total Computer Hours = $+80.0 + 105\ X_8 + 2.70\ X_{35} + 21.2\ X_{37} + .158\ X_{46} + 10.8\ X_{47..1} - 6.85\ X_{47.3} + 292\ X_{48.3} - 63.3\ X_{52} + 266\ X_{56} + 3.59\ X_{57} - 3.95\ X_{64} + 240\ X_{72} - 173\ X_{76}$

Total Months Elapsed = $1.31 + 1.31\ X_8 + 0.020\ X_{19} + .845\ X_{37} + 2.37\ X_{40} + .037\ X_{41} + .098\ X_{47.1} - .079\ X_{47.3} + .860\ X_{48.3} + 2.05\ X_{51} - 2.65\ X_{65} + 3.63\ X_{72} + .039\ X_{75} - 1.86\ X_{76}$

"+" sign in above equations show that the resource varies directly with the variable. If the variable is increased then the amount of resource will also increase. On the other hand the "-" sign represents that a resource varies inversely with variable [49]. "X" represents the dependent variables or factors that Nelson identified in his statistical experiments that finally led to the numeric values and equations. Nelson also applied these equations on the subset of total sample i-e to obtain total man months, total computer hours and total elapsed time for Business, Scientific, Computer software. Interested readers may see [49], page 80-87.
In our next sections we will use Nelson equation for computing purpose. In order to avoid the lengthy equation we adopt a general form to represent Nelson model as follows so that we may use it with ease.

$$\text{Effort} = \text{Constant value} + \sum_{i,j=1}^{14} C_i X_i$$

Another name that is more in practice for Nelson model is System Development Corporation model SDC[58][59]. However, the number of predictors used in [58] and [59] are 11 and in general form the equation may be given as.

$$\text{MM} = \sum_{c,j=0}^{11} \text{ciji}$$

$$\text{Cost} = \text{MM (Labor rate per year/12)}$$

In later sections we have used Mohanty [58] data for analysis. So for the sake of clarity, we keep both parametric equations.

3.1.4 General Research Corporation

This non-linear organizational model is developed by General Research Corporation (GRC) in 1976 to compute cost as non linear function of the delivered instructions [11][58]. GRC model estimates the development time, computer time during development and documentation time. An estimating equation is provided for analysis, design code and test phases and through this equation the effort for any phase is accomplished by using object instructions and hardware constraint factor [59]. The gernal form of GRC to estimate cost is provided by Mohanty[58] as:

$$\text{Software Cost} = 0.232(\text{LOC})^{1.43}$$

3.2. Discrete Models

These models use a tabular form that relates the effort, duration, difficulty and other cost factors [55]. Following models are grouped in this category.

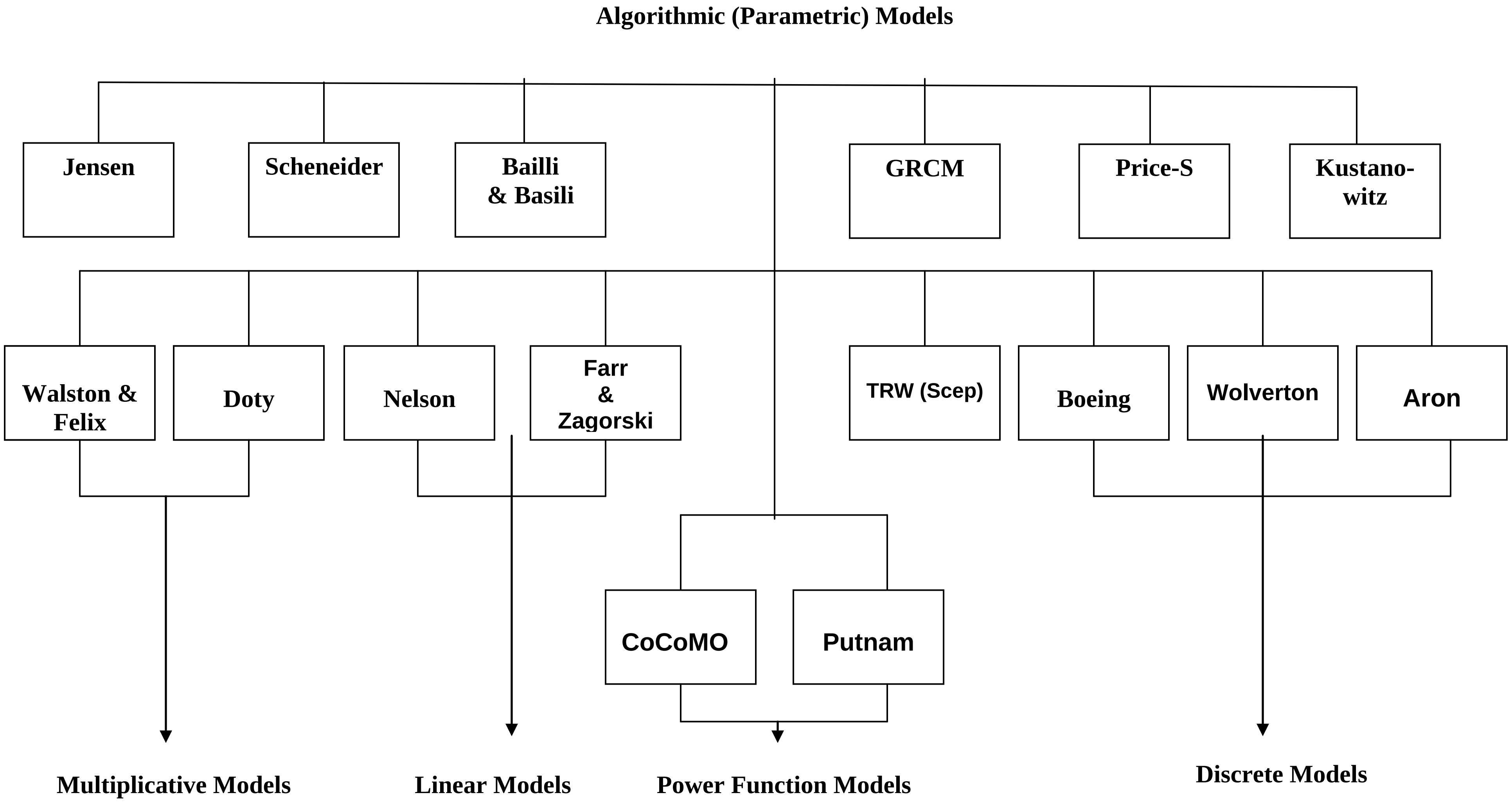
3.2.1 Boeing Model [59]

This model was developed by Boeing computer services in 1976 under the contract to US air force. The model has provided the estimates of total person-months of development, person months of development during each phase and estimates of computer time. The inputs to the model were (i) Number of lines of deliverable source code (ii) percentage of statements (iii)the six defined phases and (iv) the 8 environmental factors and each factor has six associated weight that are used to adjust and the environmental factors. (Appendix A)

The model estimates development effort for each of the six phases and amount of computer time required for development. Development time estimate is computed through percentages of statements and number of lines of code estimate. The estimate is then adjusted by environmental factors to have final estimate

Boeing model was a good effort in the sense as it was individually computed the environmental factors effecting the development phases. However, it is not widely practiced because no method for estimating life cycle, design or schedule cost was proposed by the model and the readjustments in user's environments are probably not possible.

Figure I: Algorithmic Models



3.2.2 Aron Model

Aron model was proposed by J.D. Aron that attempts to provide pragmatic determined productivity rates. Aron's model served as a basis for productivity rate estimation in software estimation field [54]. Aron model is a quantitative method of estimating manpower resources for large programming jobs. This method is based on the available historical data on productivity rate of programmers and later this data is used to adjust other non programming attributes of the system.

Aron provided four estimating methods in his paper namely Experience, Quantitative, Constraint, Units of Work [63][58].

Experience method depends upon experience with similar jobs in same environments hence providing a way for estimators to compare the system with previously completed ones. Quantitative method described by Aron[63] as "…programmer productivity in terms of the number of deliverable instructions produced per unit of time by an average programmer." Constraint method is an educated guess and units of work method is way to break the larger unit into the smaller units for which the cost is estimated with the help of available data from same historical projects. In Aron model the tasks were divided into three categories i-e easy, medium and hard. From the figures he used table IV(Appendix B), suggested that for big systems use an average productivity rate of 20 assembly language source statements per day for *easy* programs, 10 per day for *medium* programs, and 5 per day for *hard*[54].

Aron used following expression to calculate the man-months for any individual task. The total effort in man months is the sum of all individual man-months for each task.

Man Months = (Deliverable Instructions)/ (Instruction per Man Month)

Aron model is considered to be the pioneer of quantitative modeling in estimation as it has provided a good guide line for estimating man power for large systems but "…It is not being presented as a precise method and, in fact, it is not as good as an estimate based on sound experience. But it is offered as an aid to estimators working in unfamiliar areas and as a verification of estimates obtained by other methods."[63]

3.2.3 Wolverton(TRW) Model

Wolverton model was an algorithm based estimation model relied on the assumption that costs is proportional to the number of instructions. The different values that are related to the routine category and complexity level are used to convert routine size to costs. In order to divide resources among the 25 activities (Appendix B) for each of the seven phases of life cycle, a 25 x 7 matrix was used to allocate the total cost [20]. The degree of complexity or difficulty was classified as easy, medium or hard on the basis of its newness or already gone through in some previous project. Total development cost may be obtained by multiplying the cost per instruction of single routine to number of object instruction and finally summing up individual results as a single effort value. The equation used for computing effort is given as [50]

$$\text{Cost} = \sum_{m=1}^{K} \text{Effort1}(m)$$

Wolverton model also suggested the use of effort adjustment multipliers. A total of six multipliers 1) control, (2) input/output, (3) pre/post processor, (4) algorithm, (5) data management, and (6) time critical were used with the complexity scales proposed as old-easy, old-medium, old-hard, new-easy, new-medium, and new-hard[50].

Wolverton model was specifically proposed for the organizational tasks only applicable for the TRW database, that's why it was not being widely practiced in the software industry.

3.3. Multiplicative Models

These models use the coefficient values that are best for the completed project data[87]. Following models are considered to be multiplicative model.

3.3.1 Walston & Felix Model

This model was developed by Walston and Felix at IBM Federal Systems to measure the rate of production of lines of code. The model estimates the total man-months of effort as a function of the line of code to be produced [58] and also estimates pages of documentation, duration of development in calendar months, average staff size and cost of development with respect to computer time [59].

The model was a result of statistical analysis of historical data derived from a database of 60 different projects that ranged from "…4,000 to 467,000 LOC, and from 12 to 11,758 person-month effort… 28 high-level languages, and 66 computer systems and were classified as small less-complex, medium less-complex, medium complex, and large complex systems"[81].

Based on their collected data they investigated 68 variables that may affect the productivity measures. Out of those 68 variables, they selected most significant 29 factors that are associated with productivity. These factors were used to calculate the productivity index, which was computed in a linearly regression fashion to obtain an equation for estimating productivity of new projects [81][31].

Out of the nine equations used by this model, one relationship of the form $E = aL^{b}$ was used to estimate effort, where L is the number of lines of code, in thousands, and E is the total effort required in person-months. The equation obtained after deriving values for parameters a and b was [11][31].

$$\boldsymbol{E = 5.2\, L^{0.91}}$$

This model has not provided a distinction between comments and program instructions, consequently the effort for both was assumed to be same. Limited availability of this model has restricted its use or recalibrations across the organizations. The reliability of this model is questioned by different researchers and due to other weaknesses this model is probably not in practice any more.

3.3.2 Doty Model [89]

Doty associates with US air force sponsor ship incorporated this manual model in 1976/77 [11] [59]. This model is used to compute total person-months of development effort, development cost, and time, overhead cost of computer time, documentation and travel. Four application areas (i) command and control (ii) scientific (iii) business (iv) and utility are covered with the help of different equations. 14 environmental factors are also proposed in this model [table III in [72]], however their use is optional [59]. The expression used to compute effort in man Months MM for any gernal application is discussed as [72].

$MM = 5.288\ (KDSI)^{1\ 047}$
for $KDSI \geq 10$

$MM = 2.060\ (KDSI)^{1.047}$ x (effort Multipliers Fi)
for $KDSI < 10$.

Doty model has provided a way to maintain relationships for different application areas and also a way to calculate documentation and travel costs. However, Doty model was practiced only to derive estimates discovered in SDC database, so the reliability of Doty model on other environments is questioned [59]. Boehm [72] argues that Doty exhibits discontinuity at KDSi =10 and has widely varied estimates via Fi factors and adds effort estimate to 92% if the answer to factor (first development on the target computer) is Yes. The factors are given at table III (Appendix B).

3.4 Power Function Model

These models usually estimate the effort while using different cost factors and the length of code. Two mostly used models in this category are.

3.4.1 Putnam (SLIM) Model

L.H. Putnam has developed Putnam model[8][88] with the objective to recognize the resource expenditures, estimate the total software life cycle effort in person months and the time required for the development of the project. The Putnam model was developed by using a database of 40 US army computer systems along with other software data taken from 400 projects [59].

Putnam during his work observed that for the understanding of software process it will be helpful if system attribute like number of files, modules and other are related with manpower by allocating resources to different overlapping phases of life cycle, which can be characterized by Norden Rayleigh form [7]. Putnam has used this concept and assumed that the personal utilization during the development of the project is described by a Rayleigh-type curve [58] [73] [50] [31] [52] [7] and determined that the curve can be related to the number of lines of code to the time and effort required by the project. The equation proposed by Putnam is given as:

$$S \quad = \quad C_k\, K^{1/3}\, td^{4/3}$$

Where K is the total effort (in PM), S is the product size, td corresponds to the time required to develop the software. C_k is the constant value which reflects constraints on the basis of working environments. The exact value of C_k for a specific project can be computed from the historical data of the organization developing it. However, the range for C_k suggested by authors in [84] [86] [85] is between 2000 and 20,000, 6000 to 150000, and 610 to 57314 respectively. Other than this simple equation by Putnam, there were several other equations [59] proposed like cash flow equation, cumulative cost equation, life cycle cost equation design and coding equation, difficulty equation, tradeoff law, cost tradeoff equation and productivity rate equation. The readers interested in these equations can read [11] [8].

3.4.2 Jensen Model

Dr. Randall Jensen at Hughes Aircraft Co proposed Jensen model [46] to provide reasonable software estimates. The Jensen model is very alike to the Putnam SLIM model [87] discussed in above section. The Jensen proposed following equation, discussed in detail at [47] [48] [72] [87]:

$$E = 0.4(S/C_{te})^2 \times 1/T^2$$

Where *E* is the effort needed to develop the software in person years, S is effective software size, T is the development time in years and C_{te} is Jensen's technology Constant that is the slight variation in the Putnam's Constant value [87]. Technology constant in Jensen model is a product of a basic technology constant and several adjustment factors just like in the case of intermediate CoCoMo-81[87]. This value for effective technology constant can be computed by following expression [50]

$$C_{te} = C_{tb} m(X)$$

Where C_{tb} is the basic technology constant and m(X) is the cost driver adjustment multiplier (is explained in Putnam model). The cost drivers defined in *X* describe the attributes related to the product, personnel, and resource areas that affect effort [50].

Different calibrations have been made in Jensen model to achieve better performances. Authors in [84] have introduced Productivity parameter to tune the model to the capability of the organization and the difficulty of the application. They also introduced a special skills factor that varies as a function of size from 0.16 to 0.39 and enhanced the basic Jensen's equation as [84]:

$$\text{Size} = (\text{Effort}/\text{ßeta})^{1/3}\ \text{Schedule}^{4/3}\ \text{Process Productivity Parameter}$$

The model has been continually improved over the years and has provided a base for the development of commercial estimating tools like CEI JS1, JS2 and System 3 products, and the GAI SEER for Software product [48]. Jensen model was applied on classical software development projects though the results were not accurate, however, researches [45] have found Jensen model a better approach and credible as compared to Putnam model.

3.4.3 Constructive Cost Model (CoCoMo)

Cocomo by Barry Boehm in 1981 is perhaps the most complete and thoroughly documented which is practiced more than any other cost estimation model among all models for effort estimation. The reasons for the success of CoCoMo is may be its availability as an open internal public domain model or the better estimation results.

COCOMO is based on linear-least square regression with Line of Code (LOC) as unit of measure for size. Boehm proposed three levels of Cocomo namely Basic, Intermediate and Detailed [5].

Basic model is a single value static model used for computing software effort and cost as a function of program size. In condition where a rough effort estimate is desired, basic cocomo is considered effective. The general form of equation for estimating Effort, Productivity, schedule and staff for Basic cocomo is given as:

$$\text{Effort} = a(\text{KLOC})^b$$
$$\text{Productivity} = \text{KLOC}/\text{Effort}$$
$$\text{Schedule (months)} = c(\text{Effort})^d$$
$$\text{Staffing} = \text{Effort}/\text{Schedule}$$

The coefficients appeared in these equations represents the three development modes include Organic (Project is being developed in similar environment with respect to some previously developed project), Embedded (Project has hard and inflexible requirements and constraints), and Semi-Detached (Project's type falls some where between Organic and Embedded mode) [5] [61].

Intermediate Model computes effort as a function of program size and a set of Cost drivers. The equation for estimating software Effort for intermediate model slightly differs from Basic cocomo as:

$$\text{Effort} = a(\text{KLOC})^b \times m(X)$$

Where m(X) is effort adjustment factor and it is the product of 15 Effort Multipliers [5]. *a* and *b* are parameters whose values are derived from three modes as discussed above.

Detailed model include all characteristics of intermediate model with the difference that the impact of cost drivers is assessed for each and every phase of software engineering process rather for any specific phase activity. Cocomo was derived from the study of several projects and model proposed following basic equations [5].

Intermediate model can be used to make accurate predictions after the product is defined and personnel are being assigned to product development [50]. The Detail level model allows a three level hierarchical decomposition of a Software Product. Three levels are: *module*, *Subsystem*, and *system* to derive a good estimate.

The effort required to produce a Software Product can be determined using a simple overall Basic level model, a more detailed Intermediate level model, or a detailed level model, However, currently we are interested only in the equation proposed for Basic cocomo model.

The cost is usually not calculated through CoCoMo because different organizations have different productivity rates. For example, basic CoCoMo with organic mode values assuming burdened labor rate of 5000$ and for 36KLOC will yield 516771$. If average labor rate is10, 000$ then cost will equal to 1033500$.

Authors in [64] have shown that CoCoMo's results are better than Bailey & Basili, Walston & Felix and Doty model. It is also good to mention that if we use the same(LOC=36000) data provided by Mohanty [58], then CoCoMo yields following three results for three modes using basic cocomo equations.

Basic Cocomo(Organic)

Effort	=	103.35 MM
Time	=	14.6 months

Semi-detached

Effort	=	166MM
Time	=	15 months

Embedded

Effort	=	265.4MM
Time	=	15months

Author in [58] shows that for 36000 LOC, SDC effort estimate is 288.14 MM, Walston 135.6MM in 13.77 months, Aron 126.62MM, Schneider 211.42MM and doty 163.73 MM. Comparatively, it can be stated that CoCoMo with organic mode has produced better results as others of its time. Another reason to support CoCoMo is its growth in recent years with various enhancements and a number of models are developed from the basic concept of CoCoMo. These advancements resulted in the CoCoMo suit that includes the extensions of CoCoMo and independent models [65].

3.5. Others

The following algorithmic models are kept in this category on the basis that most of these were proposed and utilized by different organizations. We assembled these with no other specific reason.

3.5.1 SCEP Model [59]

TRW software cost estimating program (SCEP) was developed in 1978 to estimate person months to complete different phases (preliminary design, detailed design, coding and unit test, integration and final test) of one or more user defined subsystems.. A database of 20 completed projects [9] was used to produce different equations and determine different factors that collectively yielded development time estimates in person months.

This model was a typical work break down structure approach to assure accurate estimation and enhancing traceability. However, the limited availability and limited estimation to only few phases of life cycle has lessened the utilization of this model.

3.5.2 Kustanowitz model [91]

This model is proposed by Alvin Kustanowitz in 1977 to estimate the man power required for software implementation and this technique is famously known as SLICE (Software Lifecycle Estimation). The model proposed following stepwise approach for estimation [54].

Step 1: from the all possible activities in life cycle of software (i-e Planning Coding Feasibility Study Compilation, Conceptual Design, coding, unit test, integration test….), a list comprising a project profile which is created and modified according to the environment. According to kustanowitz a typical project profile involves 6-10 phases or steps (Functional Requirements Definition, Conceptual Systems Design, System Design, Program Specification, Coding, Unit Test, System Test). Step 2: On the available historical data, the percentages were assigned to the phases of project profile. STEP 3 & 4 : The determination of the productivity rate in the form of average number of instructions per day on the grounds of historical data and the determination that whether the productivity rate is applicable for any particular phase or not. This applicability was considered because productivity rate is determined to be LOC per day and can only be appropriate for programming activities. Step 5: Estimate the total number of instructions in the system. Step6: The estimated LOC were divided by productivity factor to get required technical man-days. The result of this division was again divided by results of step 4 to determine total number of man-days required for the project. Step 7: The determination of the manpower loading and schedule by finding the square root of the required man-months.

The productivity factors in Kustanowitz model are defined on the basis of experience, environment and programming language [58]. Authors in [92] suggested that the productivity factor for real time system is 48 to 96 instruction/ man month. [58] Used 100 as productivity factor for mathematical systems. The gernal form of the expression to compute Cost and time by Kustanowitz model is given as:

$$\text{Total Cost} = \text{MM (Labor rate per year/12)}$$

Where MM is the effort in man month which is obtained by dividing the *number of instructions* by the appropriate *productivity factor*. The expected development time in man months is the square root of Man months.

$$\textit{Expected Development Time} = (MM)^{1/2}$$

3.5.3 Schneider Model

Schneider model was proposed by Victor Schneider. This model was the outcome of the deep analysis of 400 projects that were programmed in all the major high-order languages and assembly languages taken from the open computer literature and other contractors [53]

Schneider has used Halstead's software science idea to estimate effort in man-months (MM) by the help of following equation [58].

$$MM = (2.1433 \times 10^{-7} (N\log_2 \eta)^{1.5}) / (\lambda^{0.5})$$

In this expression the program length N was $\eta\log_2(\eta/2)$**,** N's value for assembly language was given as 2667I and for FORTRAN language was 1900I. *I* was the number of instructions in thousands. The language level λ for assembly language and FORTRAN language was 0.88 and 1.14 respectively [53][58]. The values of λ and N were obtained from the statistical study of different data sets which were approved by different researchers. The following expressions [53] are used to calculate man –months.

While taking the value of $\log_2\eta = N^{.22}$ the above expression was rearranged in the following way.

$$MM =. \frac{2.1433 \times 10^{-7} (NN^{.22})^{1.5}}{\lambda^{0.5}}$$

And further to

$$MM = \frac{2.1433 \times 10^{-7} (N)^{1.83}}{\lambda^{0.5}}$$

The value of N was 2284I (i-e average of 2667I and 1900I) used in the above expression which has yielded the result as average of the effort involved in coding in HOL and in assembly language:

$$MM = 0.3I^{1.83}$$

3.5.4 Price- S Model [90]

The PRICE-S, a propriety cost estimating model was originally developed at RCA by F. Freiman and Dr. R. Park in 1977 and was used for estimating several US DoD, NASA and other government software projects [11][52][73][58]. The model was used to compute the observed factors from the historical data and then uses these factors to estimate cost, schedule, size and complexity of proposed projects [58].

Price-s also consists three sub models [73]. The Acquisition Sub model to specifically estimate the software costs and schedules. The Sizing Sub model is used for estimating the size of the software to be developed. Finally, the Life-cycle Cost Sub model is used for cost estimation of the maintenance and support phases.

In this section we have tried to provide a descriptive view of most of the famous algorithmic models. We now summarize our discussion by comparing the estimates produced by the general equations of different models for Cost, Effort and Development Time required. We use some values that are already computed by Mohanty [58]. The values which are not computed by Mohanty are computed on the basis of the assumptions in [58] that the assumed Lines of Code will be 36,000 and assumed labor rate per year is 50,000$. Table I provides the general expressions for the Effort, Cost and Development time estimation. The equations are already discussed in detail in their relevant sections, however they are provided in table I for reader's convenience. The arrangement of the models in table I are random, and the emphasis is only on the comparison of equations.

On the basis of these equations, we have computed the values for these models in Table II on the assumptions and available data, discussed above. Out of three estimated values of Wolverton model [58], we take the medium value for our comparison. From Table II we concluded that a lot of variation exists among the results of the estimation techniques for same number of LOC and burdened labor rate. These results fall in the range of 50.126 PM to 518.4 PM effort and .362 Million to 1.6 Million dollars cost. This variation is probably because of the environmental and productivity factors, different policies of the organizations that used and maintained these models, difference in the data sets selected by researchers to apply experiments, inappropriate development process, the varying currency rates, and certainly the Time itself.

Figure 2 Depicts the effort values computed from the models and figure III represents the estimated cost in millions. Figure IV represents the estimated time for development

VOL. 3, NO. 4, April 2012 ISSN 2079-8407
Journal of Emerging Trends in Computing and Information Sciences

Table I: Effort, Software Cost and Development Time Equations

Model	Effort in Man Months	Software Cost	Development Time
Farr & Zagorski	$\sum_{x,j=0}^{6} X_i j_i$	MM (Labor rate /12)	----
Bailey & Basili	0.73 * DL $^{1.16}$ + 3.5	----	----
Nelson	Constant Value+ $\sum_{i=0}^{n} c_i x_i$	----	----
Walston & Felix	5.2 $L^{0.91}$	Effort(Labor rate year/12)	2.47(Effort)$^{0.35}$
Doty	5.288 (KDSI)$^{1.047}$	Effort(Labor rate year/12)	----
GRC	----	0.232(LOC)$^{1.43}$	----
SDC	$\sum_{c,j=0}^{11} c_i j_i$	MM(Labor rate per year/12)	----
Aron	(Deliverable Instructions)/ (Instruction per Man Month)	----	----
Wolverton	----	$\sum_{M=1}^{K}$ Effort1(m)	----
Putnam	[S/C$_k$]^3 * Td$^{1/4}$ in (Person Year)	----	----
Jensen	0.4(S/C$_{te}$)^2 x 1/T^2 in (Person Year)	----	----
CoCoMo(basic)	a(KLOC)^b	Effort * Average Productivity	c(Effort)^d
Kustanowitz	----	MM (Labor rate per year/12)	(MM)$^{1/2}$
Schneider	0.3I $^{1.83}$	MM(Labor rate year/12)	----

Table II. Effort, Software Cost and Development Time results

Model	Effort in Man Months	Software Cost(Dollars in Million)	Development Time
Farr & Zagorski	86.998	.3625	----
Bailey & Basili	50.126	----	----
Nelson	119.36	----	----
Walston & Felix	135.6	0.565	13.77
Doty	163.73	0.682208	----
GRC	----	1.2245	----
SDC	288.14	1.200585	----
Aron	126.62	0.530576	----
Wolverton	----	1.13868(medium)	----
Putnam	43.2 PY or 518.4 PM*	----	----
Jensen	32.4 PY or 388.8 PM*	----	
CoCoMo(basic & Organic)	103.354	0.43064	14.56
Kustanowitz	----	1.6	19.6
Schneider	211.42	0.88092	----
Price S		1.33	18

* Technology constant taken for Putnam is 3000 and the time in year is assumed to be 2 years.
* Assumed Technology constant for Jensen model is 2000 and the time in year is assumed to be 2 years.

Figure II. Effort Estimates (Person Months-PM)

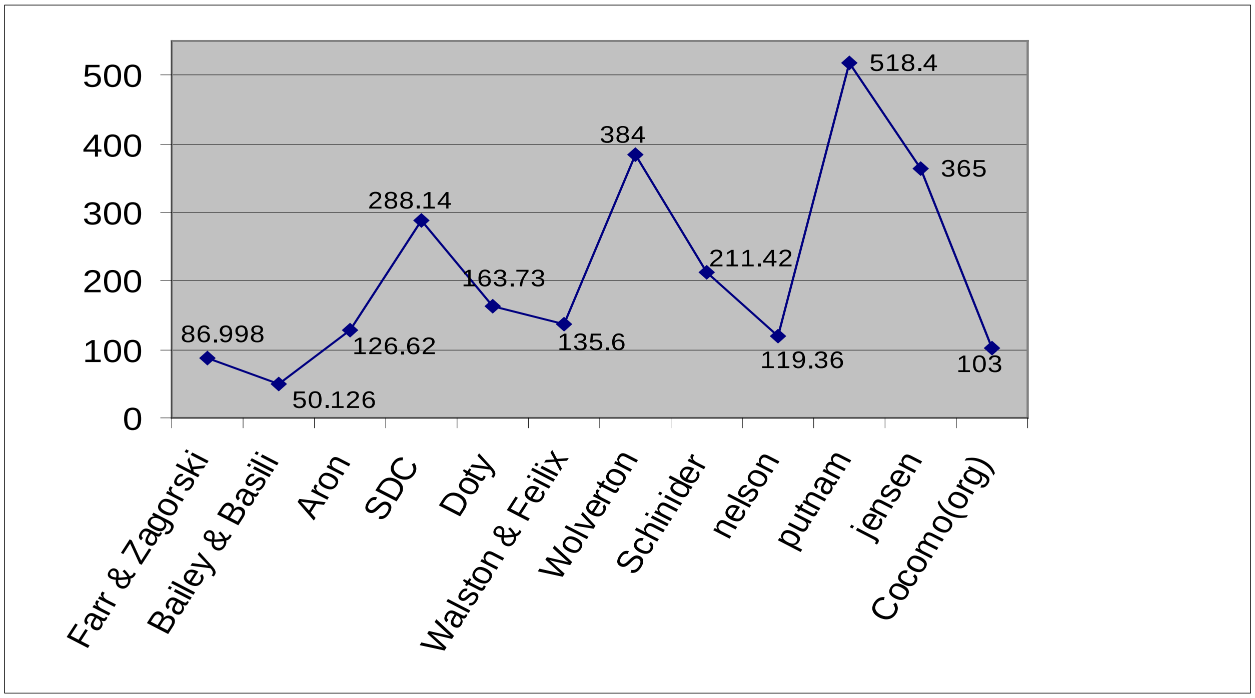
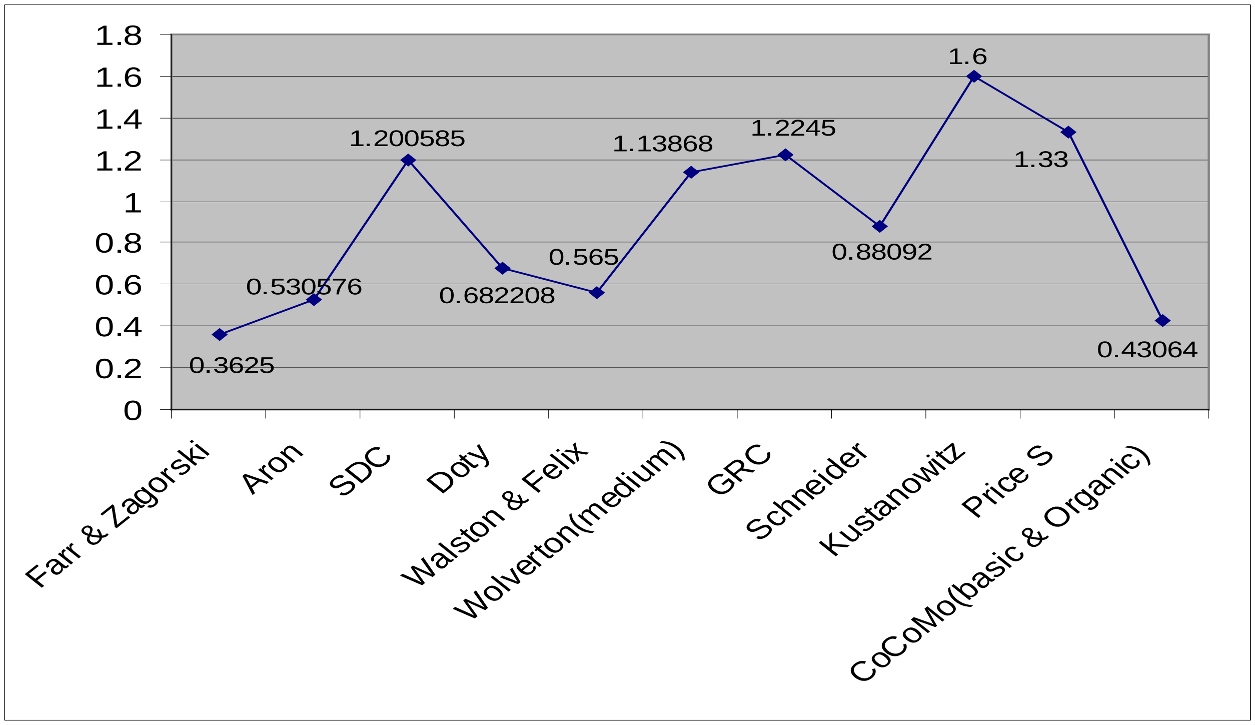


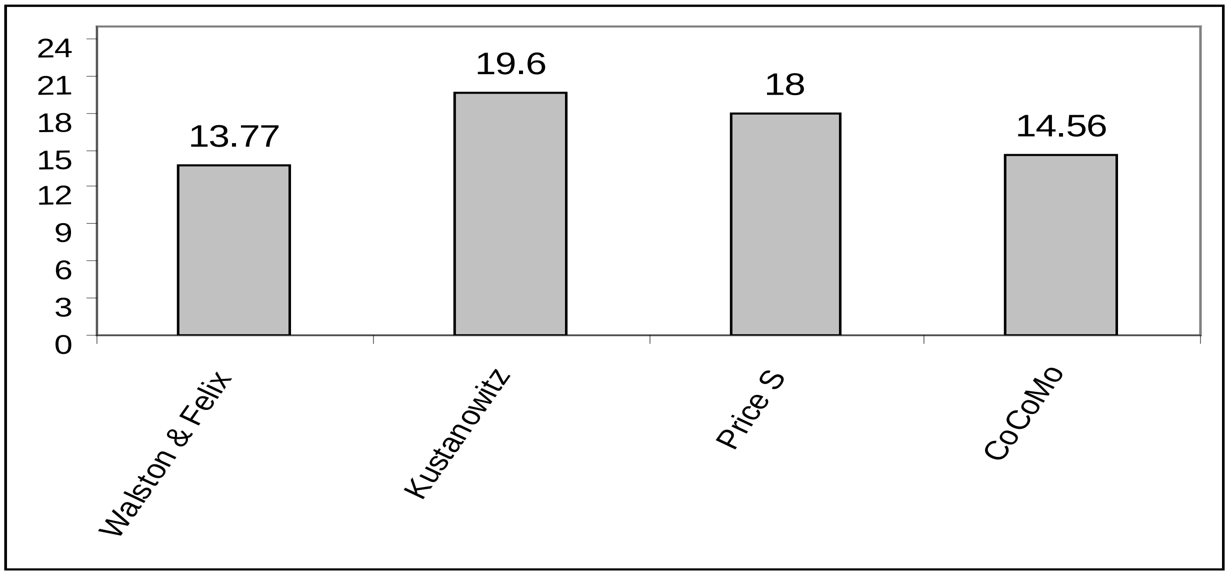
Figure III. Cost Estimates (in Million)



VOL. 3, NO. 4, April 2012 ISSN 2079-8407
Journal of Emerging Trends in Computing and Information Sciences

http://www.cisjournal.org

Figure VI. Estimated development time in Months



From all presented discussion we have seen that a lot of variation is existed in the results of algorithmic models, even computed for same LOC and burdened labor rate. At some places the result is too optimistic (Farr & Zaorski) and at some places its too pessimistic (Putnam). In such circumstances its quite hard to decide the most appropriate model. However, on the basis of discussion we recommend that the openness of CoCoMo and its continuous recalibrations makes it somehow better as compared to others of its age. Jensen model stands close in the race of enhancements and recalibrations which can be proved to be a good competitor to CoCoMos in future.

4. NON- ALGORITHMIC

In recent years the researchers have diverted their attentions toward the techniques that are more concerned with judgments and comparisons with previous work rather to use numerical models for predicting effort. Boehm[5] categorized estimation techniques into seven divisions' i-e algorithmic (above section), Expert judgment, analogy, price to win, Parkinson, top down and bottom up. In this section we will elaborate the basics of these techniques from the perspective of literature proposed on these techniques plus the trend toward the applicability of new approaches like fuzzy logic and neural networks in estimation field, however the two techniques i-e Parkinson(set the scope of the project) and price-to-win(a pricing tactic) will not be included as they are not the predicting techniques[67],

4.1 Expert Judgment

Expert Judgment sometimes known as Delphi technique is one of the most widely used Method, sometimes referred as Educated guessing technique based on intuition of some experts who make decisions during the estimation process, rather on formal models presented above. Human experts provide cost and schedule estimates based on their experience. Usually, they have knowledge of similar development efforts, and the degree of similarity is relative to their understanding of the proposed project , the guess is a sophisticated judgment supported by a variety of tools [43] The whole process of estimation could be a group discussion that ends upon a all agreed effort estimate. Boehm [5] suggests that expert judgment may be the most sensitive tool for identifying crucial differences between jobs. Prediction process highly depends upon the availability of accurate information. Different researches [44][67][42] have shown that expert judgment is not a weak techniques as compared to algorithmic models and in many cases Expert Judgment has out performed the formal techniques.

Though the Expert judgment has a number of advantages like quick production, little resources with respect to time and cost, accuracy over algorithmic techniques does not make it unbeatable. Just like other models/ techniques it has some downside like [40]:

- Subjective in nature
- One problem, different estimator will produce different estimates.

- ❖ Experience level effects estimate.
- ❖ Unstructured process
- ❖ Hard to convince customer
- ❖ Difficulty in validating estimate.

However, it may be said that Expert estimation is the method which is probably most frequently used and it can not be said that the use of formal estimation methods on average lead to more precise estimates as compared to expert judgment base method [39].

4.1.1 Top Down/Bottom up strategies for Expert Estimation

As it is mentioned earlier that expert estimation is conducted by expert persons and it is based on non-explicit, non-recoverable reasoning processes including estimation strategies supported by historical data, process guidelines and checklists. Two famous strategies for expert estimation are top-down approach and bottom-up approach [27].

Top-down strategy suggests that total effort is measurable without decomposing or breaking down the project into fewer activities or parts. The project's total effort is measured by keeping the project as a whole entity. Contrary to top-down strategy, work should be broken down into the number of activities and the effort for individual activity and the effort for each activity are estimated and the project effort becomes the sum of all individual activity's effort.

Both of these strategies can be applied while working expertly and none of both are proved a better approach when compared with each another. [26] has shown that the decomposition is a better technique while estimating effort rather to induct top-down strategy. On the other hand [25] has shown that the results obtained from bottom-up strategies are less accurate then top-down strategy.

4.2 Analogy

Analogy is a problem solving technique [37] which is used to estimate effort for a new problem by analyzing solutions that were used to solve an old problem. The parameters are searched, modified and adjusted for a current problem by a deep study and comparison from the similar cases that were already solved [30]. The analogy method usually follows the process in three step fashion [30].

(1) Selection of relevant cost attributes

The selection of relevant cost attributes may be determined by looking at the best possible combination of variables by implementing a comprehensive search by means of available clustering algorithms[29] like ANGEL [32], ACE[28] and ESTOR[33].

(2) Selection of suitable similarity/distance functions

The diversity of the projects makes their comparison difficult. In order to compare the datasets of different projects the Similarity functions are defined. Many approaches are used to assess similarity i-e the dissimilarity coefficient by Kaufman and Rousseeuw [35], Nearest Neighbor Algorithm [34], Manually Guided Induction, Template retrieval Specificity preference, Frequency preference Recency preference and Fuzzy similarity [17]

(3) Number of analogues to consider for prediction.

For prediction, one may use the effort value of the most similar analogue. When considering more than one analogue, simply the mean value (mean effort from retrieved project 1 and 2), or a weighted mean may be used for prediction.

Author in [28] suggests that analogy techniques is useful where the domain is difficult to model and can be used with partial knowledge of the target project, potential to mitigate problems with calibration and outliers and It offers the chance to learn from past experience. However, some difficulties are also faced while analogy. For conducting analogy method, four factors are required for its accuracy [28]

(i) Availability of appropriate analogue.
(ii) A sound strategy to select analogue.
(iii) Accuracy of the data used for both the analogue and the new project.
(iv) The manner whereby differences between the analogue and target are allowed for when deriving an estimate.

4.3 Fuzzy Logic and Neural Networking

The growing research in estimation fields has compelled researchers to go beyond the formal regression based models to new approaches for better estimation. These efforts have evolved the use of fuzzy approach [56] and neural network to predict effort by their application on the available formal methods.

In the estimation process it is commonly found that uncertainty exists at the beginning of projects and to determine the influence of the system's characteristic is hard to measure through metrics based on numeric values; however the use of cognitive descriptions can yield better result in an easy way [24]. This idea has given a new trend to estimation field by combining the metrics with fuzzy theory. The author in [22] discussed that the current formal models have certain problems like formal models need exact values as input, over commitment and size of data sets. The solutions suggested [22] to these problems are the set of fuzzy variables for metrics and models.

The fuzzy approach gives a range of possible values to the size of project rather to allocate the numeric values[21]. The mode can also be specified for development which is named as a fuzzy range which allows predicting effort for projects that do not fall in precise mode [16]. This predicted effort is multiplied with effort adjustment factors to yield Estimated Effort. Fuzzy logic is evolving comprehensively in the field of estimation and a number of researchers have used fuzzy logic technique to apply it on formal models [16]. For example, Jack Ryder [16] examined fuzzy techniques to COCOMO and the Function-Points models. Idri and Abran [14] applied fuzzy logic to the cost drivers of intermediate COCOMO model. Authors in [12] have used fuzzy logic to develop a metric named Fuzzy Use Case Size Point (FUSP) for the effort estimation of object-oriented software.

Another approach with close connection to fuzzy logic is emerging and that is the use of artificial neural networks to predict effort. Back propagation algorithm is the most famous technique that is used in cost estimation by Venkatachalam[93], Wittig and Finnie [94], Sarinivasan & Fisher [95] and many more. Through this technique the neurons are arranged in layers and there are only connections between neurons in one layer to the following[18]). The network generates effort while propagating cost drivers and other attributes as input through subsequent layers to final output layer [18][38]. Most recent efforts in software estimation is close collaborations of traditional estimation techniques and the use of computational intelligence techniques. Vinay Kumar et. al [96] used wavelet Nueral network for cost estimation, Pahariya et al[97] experimented a number of computational techniques in estimation, Lefley and Shepperd [98] used genetic programming to improve software estimation process, Samson[99] used Albus Perceptron method.

In this section we have assembled few non algorithmic techniques, introduced briefly some newer techniques than the ones presented in above sections. Scope of current study is however the emphasis on older techniques, therefore we avoided the in-depth discussion on Fuzzy logic, Neural networks and other computational intelligence techniques. Although the meddling of computational intelligence perhaps will enhance the accuracy of software estimation but as far as old techniques (Parametric/Non-parametric) are concerned, still most of the organization follow expertise based approaches for estimation. Every technique has its own worth in any particular scenario and these expertise based approaches cannot be neglected [44][67][42] or replaced by the mathematically proven parametric models.

5. FUTURE WORK

We have tried to cover most of the algorithmic and non algorithmic models in this paper, however, there are still few approaches that were discovered during this research, but not discussed in this paper. In future research could be conducted about all-inclusive analysis of some models in back dates like Sech-square model, Ruby Model, Daly model, Aerospace model, Kemerer model, Chen model, Goel & Okumoto model, Navel Air Development Centre model, Peters and Ramanna Model. Cost estimation process could become more reliable and sophisticated if Some new techniques that are partially stated in current study like fuzzification and Neural Networks and old algorithmic/non-algorithmic techniques in cost estimation are merge with some new approaches proposed in recent years like Genetic Algorithm for estimation by Huang and Chiu, Mantere and Alander's evolutionary methods for estimation, multivariate interpolation model by Uysal (2006). Further study can perform to check this assumption.

6. CONCLUSIONS AND SUGGESTIONS

Software Cost estimation is an important process in software development that cannot be neglected. For the proper management of any project, proper measurements must be adopted. ***"If you can't measure it, you can't manage it ~George Odiorne".*** In this paper we have summarized a number of estimation techniques/models that were proposed between 1960 and 1990, other than few techniques in non algorithmic section. The paper started with a concise overview of understanding cost estimation and factors affecting it. Software metrics were described in section 2. Algorithmic models were integrated in section 3 which were distributed into different categories. We have seen that all contributors have provided some thing new and valuable to this field of estimation. Although every approach presented in section 3 was not found perfect or without any flaw. Nearly every technique was proposed to solve the problem while working in its own environmental boundary and apparently the algorithms developed at one environment were not possibly able to be utilized at any other environment [3]. However, we have seen from the study that relatively CoCoMo is a enhanced approach than others [64] and the advancements in CoCoMo, though not discussed, are available and continuously evolving.

In section 4 we have discussed some non algorithmic approaches that are still in use and found to be more reliable then algorithmic techniques by some authors[44][67][42]. Some new approaches are presented in a bird eye view and will be discussed in details in future. We have mentioned few Computational intelligence techniques that are being practiced. However, during the study it is observed that most of the authors have conducted their experimental work on cocomo. Perhaps it would be better to enlarge the sphere of research and put some other parametric models in this race.

It is recommended that any approach alone, whether algorithmic, based on rigid mathematical proven formula or a straightforward technique just based on experience is not sufficient. Diverse techniques should be merged to estimate the effort and other factors. Expertise based techniques can be used to estimate and can be validated by the exercise of some parametric model or computational intelligence technique. Different techniques mutually will give a better chance to utilize the best attributes of those techniques.

APPENDIX A

Boeing Environmental Factors, Source [59]

1. Reimplementation of existing software
2. Follow-on contract with current custome
3. Number of programmers
4. High order language and a proven compiler
5. Macro language including forms for documentation
6. On-line data entry /coding
7. On-line debugging
8. Experience with similar applications-entry-level, moderate or high

APPENDIX B

Table III: Environmental Factors of Doty Model, Source [72]

Factors	Fi	Yes	No
1. Special displays- use of CRT displays, scopes, graphics	f1	1.11	1.00
2. Detailed definition of operational requirements	f2	1.00	1.11
3. Changes to operational requirements	f3	1.05	1.00
4. Real time operation	f4	1.33	1.00
5. CPU memory constraint	f5	1.43	1.00
6. CPU time constraint	f6	1.33	1.00
7. New CPU- is this the first software developed on the target omputer?	f7	1.92	1.00
8. Concurrent development	f8	1.82	1.00
9. Time share development environment	f9	0.83	1.00
10. Computer location	f10	1.43	1.00
11. Development at user site	f11	1.39	1.00
12. Development computer- will the development computer be different han the target computer	f12	1.25	1.00
13. Multi site development	f13	1.25	1.00
14. Computer access(programmer)	f14	Limited	1.00
		unlimited	0.90

Table IV: Productivity table used by Aron, Source [54]

Duration(Months)	6 to 12	12 to 24	more than 24	
Difficulty(Easy)	20	500	10,000	Very Few Interactions
Difficulty(medium)	10	250	5,000	Some Interactions
Difficulty(Hard)	5	125	1,500	Many Interactions
Units	Instructions Per Man-Day	Instructions Per Man-Month	Instructions Per Man-Year	

Table V: Nelson's Coefficients and Cost Drivers in 1966 SDC study, Source [49][50]

c_i	**Value**		**Cost Driver**	**Value**
c_0	-33.63	x_0	Y axis intercept	1
c_1	9.15	x_1	Lack of Requirements	0-2
c_2	10.73	x_2	Stability of Design	0-3
c_3	0.51	x_3	Percent Math Instructions	Actual percent
c_4	0.46	x_4	Percent I/O Instructions	Actual percent
c_5	0.40	x_5	Number of Subprograms	Actual number
c_6	7.28	x_6	Programming Language	0-1
c_7	-21.45	x_7	Business Application	0-1
c_8	13.5	x_8	Stand-alone program	0-1
c_9	12.35	x_9	First Program on Computer	0-1
c_{10}	58.82	x_{10}	Concurrent Hardware Development	0-1
c_{11}	30.61	x_{11}	Random Access Device Used	0-1
c_{12}	29.55	x_{12}	Different Host, Target Hardware	0-1
c_{13}	0.54	x_{13}	Number of Personnel Trips	Actual number
c_{14}	-25.20	x_{14}	Developed by Military **Organization**	0-1

Table VI: Coefficients and Cost Drivers in the Farr-Zagorski Study, Source [50]

C_i	**Value**		**Cost Driver**	**Value**
C_0	-188	x_0	Y axis intercept of the equation	1
C_1	2.86	x_1	Number of instructions	Number in thousands
C_2	2.3	x_2	Number of miles traveled	Number in thousands
C_3	33	x_3	Number of document types delivered	Actual number
C_4	-17	x_4	System programmer experience	Number in years
C_5	10	x_5	Number of display consoles	Actual number
C_6	1	x_6	Percentage of new instructions	Decimal equivalent

7. REFERENCES

[1] Z. Zia, A. Rashid, and K. U. Zaman, “Software Cost Estimation for component based fourth-generation-language software applications,” IET software, vol. 5, pp. 103-110, Feb. 2011.

[2] B. W. Boehm and P. N. Papaccio, “Understanding and controlling software costs,” IEEE Transactions on Software Engineering, vol. 14, no. 10, Oct. 1988.

[3] J. J. Bailey and V. R. Basili, "A meta-model for software development resource expenditures," in Proc. 5th Int. Conf. Software Eng., IEEE/ACM/NBS, 1981, pp. 107-116.

[4] L. Farr and J. H. Zagorski, “Quantitative Analysis of Programming Cost Factors: A Progress Report,” in Economics of Automatic Data Processing, ICC Symposium Proceedings, A.B. Frielink, (ed.), North-Holland, 1965.

[5] B. W. Boehm, Software Engineering Economics, Englewood Cliffs, NJ: Prentice-Hall, 1981.

[6] K. Lum, M. Bramble, J. Hihn, J. Hackney, M. Khorrani, and E. Monson, Handbook for Software cost Estimation, Jet Propulsion Laboratory, Pasadena, California, 2003.

[7] P. V. Norden, “On the Anatomy of Development Projects,” IRE Transactions on Engineering Management, vol. EM-7, no. 1, pp. 34-42, March 1960.

[8] L. H. Putnam, “A General Empirical Solution to the Macro Software Sizing and Estimating Problem,” IEEE Transaction on Software Engineering, vol. SE-4, no. 4, pp. 345-361, July 1978.

[9] B. W. Boehm, Early experiences in software economics, Springer-Verlag New York, Inc. New York, 2002.

[10] Y. Zheng, B. Wang, Y. Zheng, and L. Shi, “Estimation of software projects effort based on function point,” in Computer Science & Education, 2009. ICCSE '09. 4th International Conference, 2009, p. 941-943.

[11] B. Londeix, Cost estimation for software development, Addison-Wesley, 1987.

[12] M. R. Braz and S. R. Vergilio, “Using Fuzzy Theory for Effort Estimation of Object-Oriented Software,” Proc. of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI), 2004, p. 196-201.

[13] Attarzadeh, “A Novel Soft Computing Model to Increase the Accuracy of Software Development Cost Estimation,” The 2nd International Conference on Computer and Automation Engineering ICCAE, 2010, p. 603-607.

[14] A. Idri and A. Abran, “COCOMO Cost Model Using Fuzzy Logic,” 7th International Conference on Fuzzy Theory and Technology, Atlantic City, New Jersey, March 2000, p. 1-4.

[15] G. B. Ibarra, “Software Estimation Based Om use Case Size,” Software Engineering SBES Brazilian Symposium on (1979), 2010, p. 178-187.

[16] J. Ryder, “Fuzzy Modeling of Software Effort Prediction,” in proc. of IEEE Information Technology Conference, Syracuse, NY, 1998, p. 53-56.

[17] M. Shepperd and C. Schofiled, “Estimating Software Project Effort Using Analogies,” IEEE Transactions on Software Engineering, vol. 23, no.11, pp.736-743, Nov. 1997.

[18] A. Idri, T. M. Khoshgoftaar, and A. Abran, “Can neural networks be easily interpreted in software cost estimation,” in proc. of Fuzzy Systems, 2002. FUZZ-IEEE'02 the IEEE International Conference, 2002, p. 1162-1167.

[19] D. St-Pierre, M. Maya, A. Abran, J. Desharnais, and P. Bourque. Full Function Points: Counting Practice Mannaul, Technical Report 1997-04, Montreal, 1997.

[20] R. W. Wolverton, "The cost of developing large-scale software," IEEE Trans. Comput., pp. 615-636, June 1974.

[21] P. P. Reddy, K. R. Sudha, and P. R. Sree, "Application of Fuzzy Logic Approach to Software Effort Estimation," International Journal of Advanced Computer Sciences and Applications, vol. 2, no. 5, 2011.

[22] R. A. Gray and G. S. MacDonell, "Applications of Fuzzy Logic to Software Metric Models for Development Effort Estimation," in Proc .of Fuzzy Information Processing Society, 1997. NAFIPS '97., Annual Meeting of the North American, 1997, p. 394-399.

[23] C..Jones, Applied Software Measurement, Assuring Productivity and Quality, McGraw-Hill, 1997.

[24] C. Yau and R. H. L. Tsoi, "Assessing the fuzziness of general system characteristics in estimating software size," in Proc. of the 1994 Second Australian and New Zealand Conference on Intelligent Information Systems, 1994, p. 189-193.

[25] K. Moløkken, Expert estimation of Web-development effort: Individual biases and group processes (Master Thesis), in Department of Informatics. 2002, University of Oslo.

[26] T. Connolly and D. Dean, "Decomposed versus holistic estimates of effort required for software writing tasks," Management Science, vol. 43, no. 7, pp. 1029-1045, July 1997.

[27] M. Jorgensen, "Top-Down and Bottom-Up Expert Estimation of Software Development Effort," Information and Software Technology, vol. 46, no. 1, pp. 3-16, Jan. 2004.

[28] F. Walkerden and R. Jeffery, "An Empirical Study of Analogy-based Software Effort Estimation," Empirical Software Engineering, vol. 4, no. 2, pp. 135-158.

[29] M. Jørgensen, U. Indahl , and D. Sjøberg, "Software effort estimation by analogy and regression toward the mean," Journal of Systems and Software, vol. 68, no. 3, pp. 253-262, Dec. 2003.

[30] L. C. Briand and I. Wieczorek, Resource Estimation in Software Engineering in Encyclopedia of Software Engineering, Wiley Online Library, 2002.

[31] E. E. Mills, "Software Metrics," SEI Curriculum Module SEI-CM- 12-1.1, Carnegie Mellon University, Pittsburgh, 1988.

[32] M. Shepperd, C. Schofield, and B. Kitchenham, "Effort estimation using analogy," in Proc. of the 18th International Conference on Software Engineering, Berlin, Germany. 1996.

[33] T. Mukhopadhyay, S. Vicinanza, and M. J. Pietula, "Estimating the feasibility of a case-based reasoning model for software effort estimation," MIS Quarterly, vol. 16, no. 2, pp. 155–171, 1992.

[34] D.W. Aha, "Case-Based Learning Algorithms," in Proc. 1991 DARPA Case-Based Reasoning Workshop. Morgan Kaufmann, 1991.

[35] L. Kaufman and P. J. Rousseeuw, Finding Groups in Data. An Introduction to Cluster Analysis, John Wiley & Sons, Wiley Series in Probability and Mathematical Statistics, 1990.

[36] B. Kitchenham, Making Process Predictios. In Fenton, N. E Software metrics, A Rigorous Approach, Chapman and Hall 1991.

[37] S. J. Delany, P. Cunningham, and W. Wilke, "The Limits of CBR in Software Project Estimation," in Proc. the 6th German Workshop on Case-Based-Reasoning. L. Gierl, M. Lenz (eds.), Berlin, 1998.

[38] A. R. Venkatachalam, “Software Cost Estimation Using Artificial Neural Nletworks,” on Neural Networks, 1993. IJCNN '93-Nagoya. In Proc. of 1993 International Joint Conference, 1993, vol. 1, p. 987-990.

[39] K. Molokken and M. Jorgensen, “A Review of Surveys on Software Effort Estimation,” in proc. of Empirical Software Engineering, International Symposium, 2003, p.223-230.

[40] C. Rush and R. Roy, “expert judgment in cost estimating:modeling the reasoning process,” Concurrent Engineering: Research and Application (CERA), vol. 9, no. 4, pp. 271-284, 2001.

[41] V. Basili, “Models and Metrics for Software Manaement and Engineering,” IEEE. Computer Soc. Press, pp. 4-9,1980.

[42] S. Vicinanza, T. Mukhopadhyay, and M. J. Prietula, “Software effort estimation: an exploratory study of expert performance,” Information Systems Research, vol. 2 no. 4, pp. 243–262, 1991.

[43] S. L. Pfleeger, F. Wu and R. Lewis, software cost estimation and sizing methods, issues and guidelines, Rand Corporation, 2005.

[44] A.L. Lederer and J. Prasad, “A causal model for software cost estimating error,” IEEE Transactions on Software Engineering, vol. 24, no. 2, pp. 137–148, 1998.

[45] S. Conte, H. Dunsmore, and V.Y. Shen, Software Engineering Metrics and Models. Menlo Park, Calif.: Benjamin Cummings, 1986.

[46] R. W. Jensen, “A comparison of the Jensen and COCOMO schedule and cost estimation models,” in Proc. of International Society of Parametric Analysis, 1984, p. 96-106.

[47] R. Mall, Fundamentals of Software Engineering, Prentice Hall, India, 1999.

[48] SEI homepage on Jensen Model [Online].Available: http://www.seisage.net/jensen_model.htm

[49] E. A. Nelson, Management handbook for the estimation of computer programming costs, Systems Development Corporation, AD-A648750 , Oct. 1966.

[50] R. B. Gradey. (1998) CS course webpages. [Online]. Available: http://courses.cs.tamu.edu/lively/606_Simmons/Textbook/CH07_6.doc

[51] M. Iqbal and M. Rizwan, “Application of 80/20 rule in software engineering Waterfall Model,” Information and Communication Technologies, 2009, p. 223-228.

[52] Joint Government/Industry Initiative. (1995) National Aeronautics and Space administration home page on Parametric Cost Estimating handbook. [Online]. Available: http://cost.jsc.nasa.gov/pcehg.html

[53] V. Schneider, “Prediction of Software Effort and Project Duration- Four New Formulas,” ACM Sigplan, vol.13, no.6, 1978.

[54] N. B. Armstrong, “Software Estimating: A Description and Analysis of Current Methodologies with Recommendation on Appropriate Techniques For Estimating RIT Research Corporation Software Projects,” M. Eng. Thesis, Wallace Memorial Library, RIT, Dec. 1987.

[55] H. Leung and Z. Fan, Software Cost Estimation Handbook of Software Engineering, Hong Kong Polytechnic University, 2002.

[56] L. A. Zadeh, “Fuzzy Set,” Information & Control, Vol. 8, pp. 338-353, 1965.

[57] G.C.Low AND D.R. Jeffery, “Function Points in Estimation and Evaluation of the Software Process,” IEEE TRANS. SOFTWARE ENG., vol. 16, pp. 64-71, 1990.

[58] S. N. Mohanty, “Software cost estimation: Present and future,” Software—Practice and Experience Vol.11, no.2, pp.103–121, 1981.

[59] D. D. Galorath and D. J. Reifer, “Final Report: Analysis of the state of the Art of Parametric Software Cost Modeling,” software management consultants, California, Tech. Rep. Aug. 1980.

[60] J. N. M. Peeples. (2003) Miami University homepage on Farmer School of Business. [Online]. Available: http://www.sba.muohio.edu/abas/2003/vancouver/peeples_cosmo4gl%20abas03.pdf

[61] A. Caine and B. Pidduck, “f2 COCOMO: Estimating Software Project Effort and Cost, ” in Proc. of the 6th International Workshop on Economics-Driven Software Engineering Research, Scotland, May 2004.

[62] N. E. Fenton and S. L. Pfleeger, Software Metrics: A Rigorous and Practical Approach, Second Edition, International Thomson Computer Press, Boston, (1996).

[63] J. N. Buxton and B. Randell, “Software Engineering Techniques: Report on a Conference Sponsored by the NATO Science Committee,” Scientific Affairs Division, NATO, tech. rep. 27-31 Oct. 1969.

[64] K. Branley. (2004) cost estimation. [Online]. Available: http://www.branley.id.au/papers/Cost_Estimation.pdf

[65] B.W. Boehm, R. Valerdi, J. Lane, and A.W. Brown, “COCOMO Suite Methodology and Evolution,” The Journal of Defense Software Engineering, pp. 20-25, Apr. 2005.

[66] From Wikipedia, the free encyclopedia on Weighted Micro Function Points (WMFP). [Online]. Available: en.wikipedia.org/wiki/Weighted_Micro_Function_Points

[67] R. T. Hughes, “Expert Judgment as an estimating method,” information and Software Technology, vol. 38, pp. 67-75, 1996.

[68] I. Attarzadeh and O. S. Hock, “A Novel Soft Computing Model to Increase the Accuracy of Software Development Cost Estimation”. in Proc. of the 2nd International Conference on Computer and Automation Engineering (ICCAE 2010), pp. 603-607, Feb. 2010.

[69] C. Jones, Estimating Software Costs, 2nd ed., McGraw Hill, 1998.

[70] I. Attarzadeh and O. S. Hock, “A Novel Algorithmic Cost Estimation Model Based on Soft Computing Technique,” International Journal of Computer Science (IJCS), vol. 6. no. 2, pp. 117-125, Jan. 2010.

[71] A.C Hodgkinson and P.W. Garratt,” A neurofuzzy cost estimator,” in Proc. of the 3rd International Conference on Software Engineering and Applications, 1999, p. 401-406.

[72] B. W. Boehm, “Software Engineering Economics,”,IEEE trans. On soft. Eng. Vol.10, no.1, pp 7-19, 1984.

[73] B. W. Boehm, C. Abts, and S. Chulani, “ Software development cost estimation approaches-A survey,” Ann. Software Eng., vol. 10, pp. 177-205, 2000.

[74] I. Myrtveit, E. Stensrud, and M. Shepperd, “Reliability and validity in comparative studies of software prediction models,” IEEE Transactions on Software Engineering, vol. 31, no. 5, pp. 380–391, 2005.

[75] R. G. Canning, “A programmer productivity controversy,” EDP analyzer, vol. 24, no. 1, pp. 1-11. 1986.

[76] A. J. Albrecht, "Measuring application development productivity," in Share-Guide, 1979, p. 83-92.

[77] A. J. Albrecht, and J. E. Gaffney, "Software function, source lines of codes, and development effort prediction: a software science validation", IEEE Trans. software Eng., vol. se-9, pp. 639-648, 1983.

[78] P. G. Hamer and G. D. Frewin, “M.H. Halstead’s Software Science – a critical examination,” in Proc. of the 6th International

Conference on Software Engineering, Sept 1982, p. 197-206.

[79] R. S. Pressman, Software Engineering a practitioner’s Approach, McGraw Hill, 2001.

[80] W. E. Blazer, “Design and Implementation of an Intelligent Cost Estimation Model for Decision Support System Software,” M. Eng, thesis, Naval Postgraduate School, California, 1991.

[81] C. E. Walston and C. P. Felix, "A Method of Programming measurement and Estimation," IBM Systems Journal, Vol. 16, no. 1, pp. 54-73, 1977.

[82] M. H. Halstead, Elements of software science, Elsevier, New York, 1977.

[83] J. E. Matson, B. E. Barrett, and J. M. Mellichamp, “Software Development Cost Estimation Using Function Points,” IEEE Trans.Software Eng., vol. 20, no. 4, pp. 275-287, 1994.

[84] R. W. Jensen, L. H. Putnam, and W. Roetzheim, “Software Estimating Models: Three Viewpoints,” CrossTalk: The Journal of Defense Software Engineering, USA., Software Technology Support Center, Vol. 19, no. 2, February 2006.

[85] R. W. Selby, Software engineering: Barry W. Boehm's lifetime contributions to software Engineering, John Wiley & Sons, Inc., New Jersey, June 2007.

[86] C. Comstock, Z. Jiang, and J, Davies, “Economies and diseconomies of scale in software development,” *Journal of Software Maintenance and Evolution: Research and Practice,* Published in Wiley Online Library, Jan. 2011.

[87] S. D. Chulani, Bayesian Analysis of the Software Cost and Quality Models. PhD thesis, faculty of the Graduate School University of Southern California, 1999

[88] L. H. Putnam and A. Fitzsimmons, "Estimating software costs," Datamation, pp. 189-198, Sept. 1979; continued in Datamation, pp. 171-178, Oct. 1979 and pp. 137-140, Nov. 1979.

[89] J. R. Herd, J. N. Postak, W. E. Russell, and K. R. Stewart, "Software cost estimation study-Study results," Doty Associates, Inc., Rockville, MD, Final Tech. Rep. RADC-TR-77-220, vol. I, June 1977.

[90] F. R. Freiman and R. D. Park, "PRICE software model-Version 3, An overview," in *Proc. IEEE-PINY Workshop on quantitative Software Models, IEEE Cat. TH0067-9,* Oct. 1979, pp. 32-41.

[91] A. L. Kustanowitz, “System Life Cycle Estimation (SLICE): a new approach to estimating resources for application program development,” in *Proc. IEEE COMPSAC,* 1977.

[92] E. B. Daly, “Management of software development,” in *Proc. IEEE Transactions on Software Engineering,* SE-3, no. 3, 1977..

[93] A. R. Verkatachalam, ‘Software Cost Estimation using Artificial Neural Networks, *Intemational Joint Conference on Neural Networks*, *Neural Networks*, Nogoya, IEEE, 1993

[94] G. Wittig, G. Finnie, ‘Estimating Software Development Effort with connectionist Models’, *Information and* Software Technology, vol. 39, 1997.

[95] K. Srinivasan, Fisher D, “Machine Leaming Approaches to Estimating Software Development Effort”, IEEE Transaction on Software Engineering, Vol. 21, No. 2, February, 1995, pp. 126-136.

[96] K. V. Kumar, V. Ravi, M. Carr, and N. R. Kiran, “Software development cost estimation using wavelet neural Networks”, JSS., 2008, pp 1853-1867.

[97] J.S.Pahariya, V. Ravi, and M. Carr, “Software Cost Estimation using Computational Intelligence Techniques”, IEEE Transaction, 2009, PP.: 849-854

[98] M. Lefley, and M. J. Shepperd,”Using Genetic Programming to Improve Software Effort Estimation Based on General Data Sets”. Proceedings of GECCO, 2003.

[99] B. Samson, D. Ellison, and P. Dugard, “Software Cost Estimation Using Albus Perceptron (CMAC),” Information and Software Technology, 1997, vol.39, pp. 55-60.