IJITKM Volume 7 • Number 2 • Jan– June 2014 pp. 37-41 (ISSN 0973-4414)

Compendium of Software Cost and Effort Estimation Techniques

T.M.Kiran Kumar, Dr M.A.Jayaram
Department of M.C.A, Siddaganga Institute of Technology, B.H Road, Tumkur
tmkiran@yahoo.com

ABSTRACT
In Software Cost and effort estimation is a very vital task in the software industry. It involves in estimating the effort and cost in terms of money to complete the project on- time and in on- schedule. This paper gives the bird's eye view about the software effort estimation techniques which will be commonly used in the software industry. The capability to provide a good estimation on software development effort is necessitated by the project managers. Software effort estimation model divided into two main categories: algorithmic and non-algorithmic. These models too have difficulty in modelling the inherent complex relationship between the factors to find the good estimation. And this paper concludes that we cannot say which particular technique is best fit for all the situations to give an accurate estimation since cost and effort are vague. So that we make a careful comparison between all estimation approaches and choose the appropriate technique for each task. It will help us to choose which software effort estimation techniques is best

Keywords: Effort estimation, Cost estimation, Project fail, Accuracy, COCOMO.

1. INTRODUCTION

In the modest environment of Software Industry, the best organization will be the one, which the capability to develop and deliver the software product to the customers within in the promised time frame while staying in financial budgetary boundaries. Hence proper estimates are the drivers which may steer to achieve the milestones. In other words it may be said it is quiet necessary to understand and control the cost and effort by proper estimation for the good management, enhanced quality and better understanding of the software project.

The estimation is based on the historic data or the past experience. The estimating parameters are varying from the different tasks like cost, resources, manpower, technical equipment, time, schedule and other similarities between the projects are the parameters of the estimation. So the software industry is looking to produce quality products with low costs.

Software cost and effort estimation is a continuing activity which starts at the initial stage and continues through the life time of a project. Continual cost estimation is to ensure that the spending is in line with the budget.

In the last three decades, many quantitative software cost Estimation models have been developed. They range model uses data from previous projects to evaluate the current project and derives the basic formulae from analysis of the particular database available. An analytical model, on the other hand, uses formulae based on global assumptions, such as the rate at which developer solves problems and the number of problems available. Most cost models are based on the size measure, such as Lines of Code and Function Points, obtained from size estimation. The accuracy of size estimation directly impacts the accuracy of cost estimation. But none of the above leads to an accurate estimate.

2. RELATED WORK

Software project failures have been an vital subject in the last decade. Software projects usually don't fail during the implementation and most project fails are related to the planning and estimation steps. Despite going to over time and cost, approximately between 30% and 40% of the software projects are completed and the others fail (Molokken and Jorgenson, 2003). The Standish group's CHAOS reports failure rate of 70% for the software projects(Glass, 2006). Also the cost overrun has been indicated 189% in 1994 CHAOS report (Jorgensen and Molokken-Ostvold, 2006). Glass (2006) claims the reported results do not depict the real failures rate and are pessimistic. In addition, Jorgensen and Moløkken-Ostvold, (2006) indicate that the CHAOS report may be corrupted. Nevertheless the mentioned statistics show the deep crisis related to the future of the software projects. (Glass, 2006; Jorgensen and Molokken-Ostvold, 2006). During the last decade several studies have been done in term of finding the reason of the software projects failure. Galorath and Evans (2006) performed an intensive search between 2100 internet sites and found 5000 reasons for the software project failures. Among the found reasons, insufficient requirements engineering, poor planning the project, suddenly decisions at the early stages of the project and inaccurate estimations were the most important reasons. The other researches regarding the reason of project fails show that inaccurate estimation is the root factor of fail in the most software project fails (Jones, 2007; Jorgensen, 2005; Kemerer, 1987; Moløkken and Jorgensen, 2003).Despite the indicated statistics may be pessimistic, inaccurate estimation is a real problem in the software production's world which should be solved. Presenting the efficient techniques and reliable models seems required regarding the mentioned problem. The conditions of the software projects are not stable and the state is continuously changing so several methods should be presented for estimation that each method is appropriate for a special project.

3. ESTIMATION TECHNIQUES

Predominantly there are many methods for software cost estimation, which are divided into two groups: Algorithmic and Non-algorithmic. Both groups are required for performing the accurate estimation. If the needs of the project are known better, their performance will be better. In this section, we would like to discuss some popular estimation techniques which will be used in the software industry.

Algorithmic Models

These models work based on the especial algorithm. They usually need data at first and make results by using the mathematical relations. Nowadays, many software estimation methods use these models. Algorithmic Models are classified into some different models. Each algorithmic model uses an equation to do the estimation:

$$\text{Effort} = f(x1, x2\ldots xn) \qquad (1)$$

Where, (x1…xn) is the vector of the cost factors. The Differences among the existing algorithmic methods are related to choosing the cost factors and function. All cost factors using in these models are:

- Product factors: required reliability; product complexity; database size used; required reusability; documentation match to life-cycle needs;

- Computer factors: execution time constraint; main storage constraint; computer turnaround constraints; platform volatility;

- Personnel factors: analyst capability; application experience; programming capability; platform experience; language and tool experience; personnel continuity;

- Project factors: multisite development; use of software tool; required development schedule.

Quantizing the mentioned factors is very difficult to do and some of them are ignored in some software projects. In this study several algorithmic methods are considered as the most popular methods. The mentioned methods have been selected based on their reputation. There are many papers which use the selected algorithmic methods (Musilek, Pedrycz et al. 2002; Yahya, Ahmad et al. 2008; Lavazza and Garavaglia 2009; Yinhuan, Beizhan et al. 2009; Sikka, Kaur et al. 2010)

3.1 Source Line of Code

SLOC is an estimation parameter that illustrates the number of all commands and data definition but it does not include instructions such as comments, blanks, and continuation lines. This parameter is usually used as an analogy based on an approach for the estimation. After computing the SLOC for software, its amount is compared with other projects which their SLOC has been computed before, and the size of project is estimated. SLOC measures the size of project easily. After completing the project, all estimations are compared with the actual ones.

Thousand Lines of Code (KSLOC) are used for estimation in large scale. Using this metric is common in many estimation methods. SLOC Measuring seems very difficult at the early stages of the project because of the lack of information about requirements.

Since SLOC is computed based on language instructions, comparing the size of software which use different languages is too hard. Anyway, SLOC is the base of the estimation models in many complicated software estimation methods. SLOC usually is computed by considering SL as the lowest, SH as the highest and SM as the most probable size (Roger S. Pressman, 2005).

$$S = \frac{S_L + 4S_M + S_H}{6} \qquad (2)$$

3.2 Function Point Size Estimates

At first, Albrecht (1983) presented Function Point metric to measure the functionality of project. In this method, estimation is done by determination of below indicators:

- User Inputs,
- User Outputs,
- Logic files,
- Inquiries,
- Interfaces

A Complexity Degree which is between 1 and 3 is defined for each indicator. 1, 2 and 3 stand for simple, medium and complex degree respectively. Also, it is necessary to define a weight for each indicator which can be between 3 and 15.
At first, the number of each mentioned indicator should be tallied and then complexity degree and weight are multiplied by each other. Generally, the unadjusted function point count is defined as below:

$$UFC = \sum_{i=1}^{5} \sum_{j=1}^{3} N_{ij} W_{ij} \qquad (3)$$

Where Nij is the number of indicator i with complexity j and; Wij is the weight of indicator i with complexity j. According to the previous experiences, function point could be useful for software estimations because it could be computed based on requirement specification in the early stages of project. To compute the FP, UFC should be multiplied by a Technical Complexity Factor (TCF) which is obtained from the components in Table I.

TABLE I Technical Complexity Factor components

F1	Reliable back-up and recovery	F8	Data communications
F2	Distributed functions	F9	Performance
F3	Heavily used configuration	F10	Online data entry
F4	Operational ease	F11	Online update
F5	Complex interface	F12	Complex processing
F6	Reusability	F13	Installation ease
F7	Multiple sites	F14	Facilitate change

Each component can change from 0 to 5. 0 and 5 indicate that the component has no effect on the project and the component is compulsory and very important respectively. Finally, the TCF is calculated as:
TCF = 0.65+0.01(SUM (Fi)   (4)

The range of TCF is between 0.65 (if all Fi are 0) and 1.35 (if all Fi are 5). Ultimately, Function Point is computed as:
FP=UFC*TCF   (5)

3.3 Seer-Sem

SEER-SEM model has been proposed in 1980 by Galorath Inc (Galorath, 2006). Most parameters in this method are commercial and, business projects usually use SEER-SEM as their main estimation method. Software size is a key input to any estimating model and across most software parametric models. Supported sizing metrics include source lines of code (SLOC), function points, function-based sizing (FBS) and a range of other measures. They are translated for internal use into effective size (Se). Se is a form of common currency within the model and enables new, reused, and even commercial off-the-shelf code to be mixed for an integrated analysis of the software development process.

The generic calculation for Se is

Se=Newsize+ExistingSize(0.4Redesign+0.25reimp+0.35 Retest)   (6)

After computing the Se the estimated effort is calculated as below

$$t_d = D^{-0.2} * \left(\frac{S_e}{C_{te}}\right)^{0.4} \qquad (7)$$

Where D is relevant to the staffing aspects; it is determined based on the complexity degree in staffs structure. Cte is computed according to productivity and efficiency of the project method is used widely in commercial projects. (Fischman,L.This ,2005)

3.4 COCOMO

The COCOMO cost estimation model is used by thousands of software project managers, and is based on a study of hundreds of software projects. Unlike other cost estimation models, COCOMO is an open model, so all of the details are published
COCOMO-II is the latest version of COCOMO that predicts the amount of effort based on Person-Month (PM) in the software projects. It uses function point or line of code as the size metrics, Effort Multipliers and scale factors . Some rating levels are defined for scale factors including very low, low, nominal, high, very high and extra high. A quantitative value is assigned to each rating level as its weight.
COCOMO II has some special features, which distinguish it from other ones. The Usage of this method is very wide and its results usually are accurate.

3.5 Putman's model

This model has been proposed by Putman according to manpower distribution and the examination of many software projects (Kemerer,2008). The main equation for Putnam's model is:

$$S = E * (Effort)^{\frac{1}{3}} t_{d^{\frac{4}{3}}} \qquad (8)$$

where, E is the environment indicator and demonstrates the environment ability. Td is the time of delivery. Effort and S are expressed by person-year and line of code respectively. Putnam presented another formula for Effort as follows:

$$Effort = D_0 * t_{d^3} \qquad (9)$$

where, D0 , the manpower build-up factor.

4. Non Algorithmic Methods

Contrary to the Algorithmic methods, this group are based on analytical comparisons and inferences. For using the Non Algorithmic methods some information about the previous projects which are similar the under estimate project is required and usually estimation process in these methods is done according to the analysis of the previous datasets.

4.1 Estimation by Analogy

So what is analogy? Analogy is a basic human reasoning process used by almost every individual on a daily basis to solve problems based upon similar events that happened in the past. Of course analogy is not a new reasoning paradigm as it has been extensively studied and discussed by philosophers and scientists for thousands of years.
In this method, several similar completed software projects are noticed and estimation of cost and effort are compared to their actual cost and effort. By assessing the results of previous actual projects, we can estimate the cost and effort of a similar project. The steps of this method are considered as:

1. Choosing of analogy
2. Investigating similarities and differences
3. Examining of analogy quality
4. Providing the estimation

4.2 Expert judgment

Estimation based on Expert judgment is done by getting advices from experts who have extensive experiences in similar projects. This method is usually used when there is limitation in finding data and gathering requirements. Consultation is the basic issue in this method. One of the most common methods which work according to this technique is Delphi. Delphi arranges an especial meeting among the project experts and tries to achieve the true information about the project from their debates. Delphi includes some steps:

1. Coordinator gives an estimation form to each expert.
2. Each expert presents his own estimation (without discussing with others)
3. Coordinator gathers all forms and sums up them (including mean or median) on a form and tells the experts to start new iteration.
4. Steps (ii-iii) are repeated until an approval is gained.

4.3 Machine learning Models

Most techniques about software cost estimation use statistical methods, which are not able to present reason and strong results. Machine learning approaches could be appropriate at this filed because they can increase the accuracy of estimation by training rules of estimation and repeating the run cycles. Machine learning methods could be categorized into two main methods, which are explained in the next subsections.

a) Neural networks

Neural networks include several layers which each layer is composed of several elements called neuron. Neurons, by investigating the weights defined for inputs, produce the outputs. Outputs will be the actual effort, which is the main goal of estimation. Back propagation neural network is the best selection for software estimation problem because it adjusts the weights by comparing the network outputs and actual results. In addition, training is done effectively. Majority of researches on using the neural networks for software cost estimation.

b) Fuzzy Method

All systems, which work based on the fuzzy logic try to simulate human behaviour and reasoning. In many problems, which decision making is very difficult and conditions are vague, fuzzy systems are an efficient tool in such situations. This technique always supports the facts that may be ignored. There are four stages in the fuzzy approach:

Stage 1: Fuzzification: to produce trapezoidal numbers for the linguistic terms.

Stage 2: Develop the complexity matrix by producing a new linguistic term.

Stage 3: Determine the productivity rate and the attempt for the new linguistic terms.

Stage 4: Defuzzification: to determine the effort required to complete a task and to compare the existing method.

TABLE I1 Comparison of the existing methods

Method	Type	Advantages	Disadvantages
COCOMO	Algorithmic	Clear results, very common	Much data is required, It 's not suitable for any project,
Method	**Type**	**Advantages**	**Disadvantages**
Expert Judgment	Non-Algorithmic	Fast prediction, Adapt to especial projects	Its success depend on expert, Usually is done incomplete
Function Point	Algorithmic	Language free, Its results are better than SLOC	Mechanization is hard to do , quality of output are not considered
Analogy	Non-Algorithmic	Works based on actual experiences, having especial expert is not important	A lots of information about past projects is required, In some situations there are no similar project
Neural Networks	Non-Algorithmic	Consistent with unlike databases, Power of reasoning	There is no guideline for designing, The performance depends on large training data
Fuzzy	Non-Algorithmic	Training is not required, Flexibility	Hard to use, Maintaining the degree of meaningfulness is difficult

IV. CONCLUSION

The key factor for the software project failures has been the subject of many researches over a decade. According to the outcomes of several researches, the root cause for software project failures is inaccurate estimation in early stages of the project. So introducing and focusing on the estimation methods seems necessary for achieving to the accurate and reliable estimations. Since software project managers are used to select the good estimation technique based on the conditions and status of the project, describing and comprising of the project failures. There is no estimation method which can be present the best estimates in all various situations and each technique can be suitable in the special project. It is necessary understanding the principals of each estimation method to choose the best. Because performance of each estimation method depends on several parameters such as complexity of the project , duration of the project, expertise of the staff, development method and so on. Some evaluation metrics and an actual estimation example we have been presented in this paper. Making to improve the performance of the existing techniques and focusing the new methods for estimation based on today's software project requirements can be the future works in this area.

REFERENCES

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

M. E. Porter, Competitive Advantage: Creating and Sustaining Superior Performance, Free Press, NY, 1998.

R. J. Offen and R. Jeffery, "Establishing Software Measurement Programs," IEEE Software, vol. 14, 2, pp. 45-53,1997.

R. Agarwal, Manish Kumar , Yogesh, S. Mallick, RM. Bharadwaj, D. Anantwar Infosys Technologies Limited, Calcutta, India, "Estimating software projects", ACM SIGSOFT Software Engineering Notes Volume 26, Issue 4 (July 2001), Pages: 60 – 67

Barry Boehm, Chris Abts and Sunita Chulani University of Southern California, Los Angeles, USA, IBM Research, " Software development cost estimation approaches –A survey". Annals of Software Engineering volume 10, issue 1-4 (2000) pages 177–205, Year of publication: 2000 ISSN: 1022-7091

Hareton Leung, Zhang Fan, "Software Cost estimation", Hong Kong Polytechnic University, 2002

C. Ravindranath Pandian, Software Metrics A Guide to planning, Analysis and Application, India, 2004

A. Albrecht, "Measuring Application Development Productivity", in Proceedings of Joint SHARE/GUIDE/IBM Application Development Symposium, October 1979.

A. Abran, P.N. Robillard, "Function point analysis: an empirical study of its measurement process," IEEE Trans. Software Eng., vol. 22, 1996, pp.895-909.

Swapna Kishore, Rajesh Naik, **"**Software Requirements And Estimation**"**, McGraw-Hill, India, 2005

Roger S. Pressman**, "**Software Engineering, A Practitioner's Approach**"** Sixth Edition, McGraw-Hill, NY, 2005.

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

S.A.Whitmire,"3D function points" scientific and real-time extensions to function points," in Proceedings of the 1992 Pacific Northwest Software Quality Conference, 1992.

C.R. Symons**, "**Function point analysis: difficulties and improvements" IEEE Trans. Software Eng**.**, vol. 14, no.1, 1988, pp. 2-11.

Symons, "Software Sizing and Estimating – Mark II FPA," Symons C., John Wiley and Sons, U.K., ISBN 0-471-92985-9, 1991.

Desharnais, J.-M.; St-Pierre, D. Maya, M. Abran, A. "Full Function Points: Counting Practices Manual - Rules and Procedures", Montréal, Université du Québec à Montréal , 1997.

C.R. Symons and P.G. Rule, "One size fits all- COSMIC aims, design principles and progress", in Proceedings of 10th Conference on European Software Control and Metrics, 1999, pp. 197-207.

B. W. Boehm, Software engineering economics, Englewood Cliffs, NJ: Prentice-Hall, 1981.

B.W. Boehm et al "The COCOMO 2.0 Software Cost Estimation Model", American Programmer, July 1996,

Robert T. Futrell, Donald F. Shafer, Linda I. Shafer, "Quality Software Project Management**"**, Software Quality Institute Series, 2004

Ian Sommerville, "Software Engineering, 8th edition**"**, 2009

L. H. Putnam, "A general empirical solution to the macro software sizing and estimating problem**"**, IEEE Trans. Soft. Eng., July 1978, pp. 345-361.

Watts Humphrey, A Discipline for Software Engineering,Addison Wesley, 1995

Prasad Reddy, (2010), "Particle Swarm Optimization in the fine-tuning of Fuzzy Software Cost Estimation Models, International Journal of Software Engineering (IJSE), Volume (1): Issue (1), pp 12-23.

Prasad Reddy P.V.G.D, Sudha K.R, Rama Sree P and Ramesh S.N.S.V.S.C, (2010), "Software Effort Estimation using Radial Basis and Generalized Regression Neural Networks", Journal of Computing, Volume 2, Issue 5, pp 87-92.

Razaz, M. and King, J. (2004) "Introduction to Fuzzy Logic" Information Systems - Signal and Image ProcessingGroup. http://www.sys.uea.ac.uk/king/restricted/boards/

S. Kumar, B. A. Krishna, and P. S. Satsangi, (1994), "Fuzzy systems and neural networks" in software engineering project management, Journal of Applied Intelligence, no. 4, pp. 31-52.

Sun-Jen Huang and Nan-Hsing Chiu, (2007), "Applying fuzzy neural network to estimate software development effort", journal of Applied Intelligence. Vol 30 Issue 2, pp.73-83

Urkola Leire , Dolado J. Javier , Fernandez Luis and Otero M. Carmen , (2002), "Software Effort Estimation: the Elusive Goal in Project Management", International Conference on Enterprise Information Systems.

X. Huang, J. Ren and L.F. Capretz, (2004), "A Neuro-Fuzzy. Tool for Software Estimation", Proceedings of the 20th IEEE International Conference on Software Maintenance, pp. 520.

Xu, Z. and Khoshgoftaar, T. M., (2003), "Identification of fuzzy models of software cost estimation".