COMPARATIVE ANALYSIS OF SOFTWARE ESTIMATION MODELS

[1]Dr. AM NAGESWARA YOGI, [2]Mr. SUNIL KUMAR M, [3]Mr. CHANDRASHEKAR P
[1]Professor & HOD, [2]Associate Professor, [3]Assistant Professor,GFGC, Nelamangala,Bangalore Rural.
Department of MCA, T.John Institute of Technology, Bangalore – 560083.

Abstract - **Systematic software development process involves estimation of size, effort, schedule and cost of a software project and analysis of critical factors affecting these estimations. These are the crucial factors dictating the project manager whether to take up the development of software project or not. In literature we find that majority of the software projects have failed. Failed projects are the ones which have challenged the initial estimates and not meeting the user requirements. Therefore, before biding for a software project the project manager has to spend sufficient time and energy for software estimation and risk analysis. In the last five decades a number of software estimation models have been developed. However different methods give different effort values for the same input. This is because every model is developed for particular environment factors under which the project was developed. The project manager will be confused in selecting a model for his/her project since environment under which the project to be taken up will be entirely different from those of the previous projects. A project manager will be successful, if he or she will be able to find reasons for different effort values given by different methods. In this paper we have studied a few software estimation models and compared them after obtaining results for four software projects.**

Keywords: Software, Project, Effort, Estimation, Performance.

I. INTRODUCTION

Projects are relevant today to all fields and sectors. A project which can be broadly defined as set of activities which have a defined start state and a defined end state and which pursue a defined goal and use a defined set of resources come in many various form and in almost any area. There may not be any organization including academic institution in the world which is not project based. From Table 1 below [1] we observe that success rate of software projects is around 40% during 2012 whereas it was just 16% during 1994. The improvement in the success rate is due to development of better software estimation and risk methodologies during the last two decades. However, as on today there are no universally applicable estimation and risk management techniques since the environment and technologies are different for different software projects.

A project is said to be successful if it is delivered within the estimated budget and time and meets all the specified user requirements. When we analyze the cause for such large failure rate, we basically identify that software estimation and risks are the factors contributing to failure rate of software projects. To overcome from such a precarious condition, we have to study critically the role of project managers in development of successful software projects both in estimation and risk management. In this paper we try to discuss and analyze basic estimation models and study the performance of these models for four case studies (software projects.

TABLE 1
SOFTWARE PROJECTS SUCCESS OVER THE LAST 20 YEARS

YEAR	1994	2000	2004	2006	2008	2010	2012
Successful	16%	28%	29%	35%	32%	37%	39%
Failed	31%	23%	18%	19%	24%	21%	21%
Challenged	53%	49%	53%	46%	44%	42%	43%

II. EFFORT ESTIMATION MODELS

Typical estimation models are derived using regression analysis on data collected from past software projects. The overall structure of such models takes the form:

$$E = A + B\ (SIZE)^{C},$$

Where A, B, and C are empirically derived constants, E is the effort in person-months (PM) and SIZE is the estimation variable either in KLOC (Kilo Lines Of Code) or FP (Function Points). Most of the models use adjustment factors to cater for other project characteristics. The constants derived for the following KLOC-oriented models [2] for which SIZE = KLOC are as follows and are briefly discussed in this Section:

A. Boehm simple model A = 0.0, B = 3.20, C = 1.05
B. Bailey-Basili model A = 5.5, B = 0.73, C = 1.16
C. Doty model A = 0.0, B = 5.288, C = 1.047
D. Walston- Felix model A = 0.0, B = 5.2, C = 0.91

In addition following two software estimation models based object points are also discussed.

E. COCOMO II model,
F. Yogi - Patil model.

A. Boehm simple model

The basic COCOMO model by Dr. Boehm is a static, single-valued model that computes software development effort (cost) as a function of program size expressed in estimated KLOC. Nominal Effort (NE) in person-months (PM) and Development Time (DT) are expressed as exponential functions of KLOC as follows:

$$NE = a\,[KLOC]^b \text{ Person months,}$$
$$DT = c\,[NE]^d,$$

where a, b, c and d are constants to be determined using the software characteristics such as complexity, skill level of team members etc. The cost of the software project may then be estimated based on the cost per person-month, which vary from a country to another.

B. The Bailey-Basili Model

John Bailey and Victor Basili attempted to present a model generation process for developing a local resource estimation model, which consisted of following three steps [3]:

- Compute background equation,
- Determine factors explaining the differences between actual project data and the mean of the estimates derived from the background equation,
- Use model to predict new project.

The background equation or the baseline relationship between effort (E) and size (KLOC) was determined using 18 data points from the NASA SEL (Software Engineering Lab) database and is given by

$$E = 0.73 * (KLOC)^{1.16} + 5.5$$

This equation can be used to predict the effort required for an average project. The next step in the process is to determine a set of factors that differentiates one project from another and helps to explain the difference between actual efforts versus efforts estimated by the background equation.

C. The Doty Model

This model is the result of an extensive data analysis activity including many data points from Software Development Corporation sample sets. The model for general application is

$E = 5.288\,(KLOC)^{1.047}$, for KLOC > 10.

$$E = 2.060\,(KLOC)^{1.047}\left(\prod_{j=1}^{14} fi\right), \quad \text{for KLOC} \le 10.$$

The effort multipliers f_i are given in Table 2.

D. Walston and Felix model

Walston and Felix [4] analyzed data of more than 60 projects done at IBM Federal Systems Division, ranging from 4K to 467K lines of delivered source code and found that if the size estimate is in thousands of delivered lines of code (KLOC), the total effort E in person-months (PM) can be given by

$$E = 5.2\,(Size)^{0.91}.$$

E. COCOMO II application composition

COCOMO II application composition model uses Object Points as a basic parameter. Object Point is an indirect software measure that is computed using counts of the number of Screens (at the user interface), Reports to be generated and 3GL components likely to be required to build applications. Each object point is classified into one of three complexity levels i.e. Simple, Medium and Difficult using the criteria suggested by Boehm [5]. In essence complexity is a function of the number and source of the client and server data tables that are required to be generated, the screen or report and the number of views or sections presented as part of the screen or report. Once the complexity is fixed, the number of screens, reports and components are weighted according to values given in Table 3.

TABLE 2
DOTY MODEL PARAMETERS FOR KLOC<10

Factors	f_i	Yes	No
Special display	f_1	1.11	1.00
Detailed definition of operational Requirements	f_2	1.00	1.11
Change to the operational requirements	f_3	1.05	1.00
Real-time operation	f_4	1.33	1.00
CPU memory, constraints	f_5	1.43	1.00
CPU time constraints	f_6	1.33	1.00
First software developed on CPU	f_7	1.92	1.00
Concurrent development of ACP hardware	f_8	1.82	1.00
Time share versus batch processing in development	f_9	0.83	1.00
Development using computer, at another facility	f_{10}	1.43	1.00
Development at operational site	f_{11}	1.38	1.00
Development computer different than target computer	f_{12}	1.25	1.00
Development at more than one site	f_{13}	-	1.00
Programmer access to computer	f_{14}	-	0.90

TABLE 3
COMPLEXITY WEIGHTS FOR OBJECT TYPES

Object type	Complexity weight		
	Simple	Medium	Difficult
Screen	1	2	3
Reports	2	3	8
3GL Component	-	-	10

The Object Points (OP) count is then determined by multiplying the original number of object instances by the weighting factor and summing to obtain total object point count. When component-based development or general software reuse is to be applied the percent of reuse (%reuse) is estimated and the object point is adjusted.

New Object Points (NOP) = OP [(100 -%reuse)/ 100]

(3.6)

To derive an estimate of effort based on computed NOP value a "productivity rate" must be derived. Productivity rates are given in the Table 4.

Productivity rate (PROD) = NOP/ person-month

TABLE4
PRODUCTIVITY RATE FOR OBJECT POINTS

Developer's Experience/ Capability	Very low	Low	Nominal	High	Very High
Environment maturity / Capability	Very low	Low	Nominal	High	Very High
PROD	4	7	13	25	50

The above equation is used for different levels of developer experience and development environment maturity. Once the productivity rate is determined, an estimate of project effort is computed using

Estimated effort (PM) = NOP/PROD

F. Yogi - Patil effort estimation model

Good software cost models can significantly help software project managers. With good models, project stakeholders can

make informed decisions about how to manage resources? How to control and plan the project? How to deliver the project on time on schedule and on budget? Therefore, project manager needs ways to address three challenges of today. First, he needs to measure projects and processes. To reduce measurement costs, he needs to collect and analyze data based on the goals of the project. Second he should use updated estimation models as and when they become available. The project manager should use his own metrics built using local data and environment to formulate simple estimation models that are better suited for the types of new projects, development process and the environment under the project has to be developed. The project manager should never rely on a single model for estimation because all models are only approximations. Third, the project manger must be prepared to change metrics and models as he or she gains new understandings of the new development processes, technologies and tools. This will continue to be a very dynamic and exciting area for research. Keeping this in our mind a methodology for effort estimation for software projects to be developed under 4GL environment was formulated and published as Yogi - Patil method [6] based on local data and local working conditions. Assumptions made in formulating the method were based on the environment under which we worked for the last three decades.

Basic idea behind this method was from Boehm's COCOMO II model. In Yogi - Patil effort estimation model, instead of Object Points of COCOMO II, we consider number of screens (front-end forms) to be designed and developed for the project. Each front-end form is assigned Design Complexity level by defining two parameters -- Event Complexity (EC) and Interactivity level (IL). The EC and IL are measured on five point scale as Very Simple (VS), Simple(S), Average (A), Moderate (M) and Complex (C)/ Extensive (E). After this effort unit matrix (See Table 5) was formulated which gives effort units required to design and develop each front-end form whose EC and IL complexity is given. Before arriving at this matrix a number of case studies were taken and partial shooting method was used to arrive at these factors. Suggestions were also invited from 15 project heads from software industry. The entire analysis was placed before a number of experts for their opinion.

TABLE 5
EFFORT UNIT MATRIX

IL	EC				
	VS	S	A	M	C
VS	2.5	5.0	7.5	10	12.5
S	5.0	7.5	10	12.5	15
A	7.5	10	12.5	15	17.5
M	10	12.5	15	17.5	20.
E	12.5	15	17.5	20	22.5

The derived effort unit matrix for 4GL forms is a 5 X 5 symmetric matrix, in which a row depicts a situation of increasing effort as EC increases. The column depicts a situation for a form of a specified event complexity level, increasing effort as IL increases. The first element in the above matrix indicates that the effort required for designing, developing and integrating (with other forms, reports and database) a form with VS-IL and VS-EC is 2.5 effort units which is assumed to be equal to the work done in 20 person-hours. We represent the element of this table as b_{ij}, i=1.. 5 and j=1,.. 5.

Based on the requirements specifications of the software project, number of front-end GUIs forms required to be design and developed can be estimated. These forms were classified into various levels of EC and IL as follows: 10 percent of the total number of front-end forms estimated were assumed to be of the type VS-EC and VS-IL, 20 percent of the total number of forms were assumed to be of the type S-EC and S-IC, 40 percent of the total number of forms were assumed to be of the type A-EC and A-IL. Again 20 % and 10% of the total number of forms were assumed to be of the type M-EC and M-IL, C-EC and E- IL respectively. This assumption is validated and verified for 30 software projects. The classified number of forms are again arranged in 5X5 matrix whose elements were denoted as a_{ij}, i=1,..5 and j=1,..5. As the project progress, clearly the number of front-end forms to be developed to complete the project emerges more and more accurately. Then effort required to design and develop all the front-end forms is calculated as:

Forms_Effort (in TEU) = $\sum_{i,j=1}^{5} a_{ij} b_{ji}$, where TEU is the total effort units. Effort required for creation of database, reports to be generated, documentation and networking requirements (if any) are calculated separately [6,7] and are added for obtaining total effort as per the following formula:

Total effort units (TEU) required for developing a software project = Forms_Effort + Db_Effort + Rp_Effort + Nw_Effort +Doc_Effort,

where

Db_Effort = effort required to create and link with the software,

Rp_Effort = effort estimate to design and generate reports.

Nw_Effort = effort estimated for making the software to work on network,

Doc_Effort = effort required for producing required documents.

Underlying assumptions, concepts and process involved in the Yogi - Patil methodology are discussed in [6, 7].

III. RESULTS AND DISCUSSIONS

Four software projects with size 22, 8, 5, 1.5 KLOC have taken up for comparison and estimation of efforts values from estimation methodologies discussed in the previous section. One can look into details about these case studies in [8]. The effort values obtained are tabulated in Table 6. In this table methods 1, 2, 3, 4, 5 and 6 refers to Boehm simple model, Bailey-Basili, Doty, Walston-Felix, COCOMO II and Yogi – Patil models respectively

TABLE 6
ACTUAL AND ESTIMATED EFFORT (E_i) IN PERSON-MONTHS

Methods→ Case Study↓	1	2	3	4	5	6	KLOC	Actual effort
I	83	32	135	87	77	64	22	77
II	30	14	48	36	30	50	8	60
III	19	11	30	24	19	31	5.	42
IV	5	7	9	8	6	13	1.5	12

Here we consider that the case study I as a large project, case studies II and III as medium sized projects and the case study IV as a small project. Ratio of actual effort to estimated effort is an important parameter for academic interest. This ratio will give an idea that how much estimated values differs from the actual value? These ratios are calculated and are given in Table 7.

TABLE 7
RATIO OF ACTUAL EFFORT TO ESTIMATED EFFORT

Methods→ Case Study↓	1	2	3	4	5	6	Actual effort
I	0.93	2.44	0.57	0.88	1.01	1.2	1.0
II	2.06	4.34	1.26	1.7	2.01	1.2	1.0
III	2.31	4.01	1.4	1.79	2.32	1.35	1.0
IV	2.40	1.88	1.48	1.6	2.30	0.92	1.0

It can be observed that Boehm simple, COCOMO II, Walston-Felix methodologies give effort values that are close to actual effort in respect of case study I. For other case studies the actual effort is around 2 to 2.4 times the values obtained by Boehm simple and COCOMO II and around 1.7 times the values obtained by Walston-Felix. Therefore, we conclude that these methods give good estimates for large projects like case study I. Bailey-Basili gives the effort values that are very different from the actual effort for medium size projects like II and III. The values obtained by Doty method for II, III and IV are reasonably closer to the actual values, but for case study I this method gives 2.44 times the actual effort. Walston-Felix, and Yogi - Patil methods give uniform trend for ratio of the actual effort to the estimated effort for all the sizes of the project. We observe that Yogi - Patil method gives comparable estimates with the actual effort values for all types of projects like large, medium sized, smaller sized projects.

IV. PERFORMANCE OF THE ESTIMATION MODELS

There are various measures for assessing the accuracy of the models: Root Mean Square Error (RMSE), Mean Relative Error (MRE) and so on. RMSE is calculated based on a number of actual data observed and estimated values by a model; it is derived from basic magnitude of relative error which is defined as

$$RMSE = \sqrt{1/N \sum (E_a - E_i)^2} \quad ,i = 1, N$$

where, E_a is the actual effort and E_i is the effort estimated by a selected method for i^{th} case study and N is the number of case studies or number of observations. Mean Relative Error (MRE) is the main performance measure. It is estimated from relative error, which is the relative size of the difference between the actual and estimated value. MRE is the percentage of the absolute values of the relative errors averaged over the number of observations. Here number of observations is again the number of case studies worked out. Hence,

$$MRE = 100 * (1/N) \sum (|E_a - E_i|) / E_i$$

Based on the effort values obtained by the Yogi-Patil method and from other methods (Table 5), RMSE and MRE were estimated from the above equations respectively for each of the methods and the values are tabulated in Table 7 for analyzing the performance of all the methods.

Table 8: MRE and RMSE for methods

Method →	Barry Boehm	Bailey -Basili	Doty	Waslto n-Felix	COCO MO-II	Yogi – Patil
MRE	92	206	35	51	90	21
RMSE	19	36	30	16	20	10

The results from Table 7 shows that the Yogi - Patil method has the lowest MRE and RMSE values i.e. 20.9 and 9.89 respectively. Hence the methodology formulated was able to provide reasonably good estimates.

V. ADVANTAGES OF YOGI - PATIL METHODOLOGY

The main advantages with Yogi - Patil methodology are:

1. It is difficult to estimate the size of the project in terms of KLOC during the initial stage of software project. However for those methods which needs KLOC as an input for software estimation, the project manager approximately estimates the size and has to multiply by various factors representing environment, technological changes etc. Yogi – Patil method is not dependent on KLOC and hence avoids the approximations.
2. Based on the requirements specifications it is easy guess how many front-end GUIs forms are required to be design and developed, which one of the input for this method. Classification of forms is not difficult.
3. The method is simple and easy to use and does not need advanced mathematical knowledge.
4. The event complexity and interactivity level of GUI screens and the reports can determined easily using the number commands and other controls that are going to be placed on the screen.
5. It is very easy to repeat the calculations as many number of times.

VI. DISADVANTAGE OF YOGI-PATIL METHODOLOGY

The logic behind formulation of effort unit matrix and classification of the design complexity of the front-end forms for effort estimation were formulated based on expert opinion and experience of the project managers. Therefore, some percentage of error in judgment in formulating this matrix and classification cannot be avoided.

VII. CONCLUSIONS

In this paper we have discussed some of the basic effort estimation techniques. We have taken four software projects whose size varied from 1.5 to 22 KLOC for effort estimation from all the discussed methods. We observe that Yogi - Patil methodology gives effort values close to the actual effort value. The evaluation of MRE and RMSE also has indicated the Yogi-Patil method gives effort values which are comparable to the actual effort values. The main reasons for this are that the method was developed keeping working conditions like number of hours we work in a day/ month/ year, the skill level of our people and various other factors considered related to Indian environment. The project manager is required to estimate the software project using at least four methods and investigate the reasons why different

methods give different results. If he/she is able to find the reasons behind the variations, then it will be easy for the project manager to select the best suitable method for software estimation.

ACKNOWLEDGEMENTS

Authors wish to acknowledge Dr. Thomas P John, Chairman, T.John Group of Institutions and Dr. S Padmanabha, Principal, T.John Institute of Technology for their encouragement and constant support during the investigations that are recorded in this paper. Also authors thank them for their permission to present this paper in the conference.

REFERENCES

[1] www.standishgroup.com accessed on 15-02-2014.

[2] Pressman R. S, Software Engineering A Practitioner's Approach, Seventh Edition, Tata McGraw-Hill, 2010.

[3] Sunita Chulani , Bayesian Analysis of Software Cost and Quality Models, A Thesis, 1999

[4] Walston C. E. and Felix C. P, "A method of programming measurement and estimation", IBM Systems Journal, Vol. 16, No. 1 pp. 54-73, 1977.

[5] Boehm B. W, "Anchoring the software process", IEEE Software, Vol.13, No.4, pp.73-82, 1996.

[6] Yogi A. M. N and Mala,V. Patil, "Software Effort Estimation Models and Performance Analysis with Case Studies", International Journal of Computer Applications in Engineering, Technology and Sciences, Vol. 1, No. 2, pp. 558-565, Apr-Sep. 2009.

[7] Mala V Patil and AMN Yogi, "Importance of data collection and validation for systematic software development process", Int. J. of Computer Science & Information Technology, Vol.3, No.2, pp. 260-278 Apr. 2011.

[8] Mala C Patil. "Estimation methodologies and risks analysis for success of software projects." Ph. D Thesis, Anna University, Chennai, 2013.