International Journal of Computer Applications (0975 – 8887)
Volume 11– No.9, December 2010

Analysis of Productivity Gain in Incremental Effort Estimation

Devesh Kumar Srivastava
Department of CSE
Uttrakhand Technical University
Dehradun India

Durg Singh Chauhan
Department of CSE
Uttrakhand Technical University
Dehradun India

ABSTRACT
This paper presents the framework for incremental Effort based development in order to analysis the productivity gain in Effort based development. Effort estimation is a challenge in every software project. The estimates will impact costs and expectations on schedule, functionality and quality. While expert estimates are widely used, they are difficult to analyze and the estimation quality depends on the experience of experts from similar projects. Alternatively, more formal estimation models can be used. Traditionally, software size estimated in the number of Source Lines of Code (SLOC), Function Points (FP) and Object Points (OP) are used as input to these models. Models that predict product size as an exponential function of the development effort are used in the paper to explore the relationships between effort and the number of increments. The author mainly focus what will be effect on productivity rate on incremental development and how duration for incremental software development vary .For incremental development the author estimate the cumulative effort gain against effort estimation . This research paper will be helpful to get productivity rate against incremental effort estimation.

General Terms
Effort estimation, Productivity, Time duration.

Keywords
Kilo line of source code, Estimation models. Effort, Business Process outsourcing (BPO).

1. INTRODUCTION
Effort estimation of software is the preliminary phase between the client and the business enterprise. Still there is a challenge for project managers in software engineering to estimate the actual Effort which meets to standard as per company policy. Today there are so many software available in global market, which requires updating as the requirement of client's demands changing from time to time. The fast changes can be seen in BPO industry for that software which is used for calling. Accurate estimation is a complex process because it can be visualized as software Effort prediction. The aim of this paper is to provide guidance for other organizations that want to improve their estimation process. The market is fast-moving characterized by a dynamic environment with high levels of uncertainty and risks. Customers appear more demanding and discerning expecting non-stop service around the clock. Service is being judged according to overall traffic, frequency and duration of visits and loyalty. The scope is highly changeable over time to respond to competition and opportunities. Therefore time becomes the critical factor today. As the speed of release is measured by getting there before the competition, project execution is measured according to the shortest time to register a presence (often regardless of the quality). Trade-offs can often sacrifice scope, cost, expectations or quality to accelerate the speed of completion. Most projects in this environment are new, innovative and difficult to estimate and cost. The planning approach thus needs to focus on key milestones and targets, yet remain flexible enough and responsive enough to cope with changing requirements, delivery dates, release deadlines and new opportunities. Incremental and iterative development approaches have long been recognized as effective in reducing the risk of failure in such situations as they entail a more controlled approach to development. Rapid Application Development and agile methods are particularly useful in environments that change regularly and impose demands of early solutions. The motivation for this work is thus derived from the current interest in speeding up development schedules. A key implication of the shift to more rapid development methods is the growing emphasis on fixed time and fixed Effort in projects. Going back to the triple constraint, if time schedules are fixed by time boxes and costs are largely dictated by the availability of personnel to work within these imposed time frames, the main variable is the scope that can be delivered.

2. SIZE OF THE DEVELOPMENT EFFORT
Estimating the development Effort for software system is a long standing problem in software project management. It has generally been noted that the Effort (*E*) is strongly correlated to the program size. A great deal of research has been carried out to relate the Effort empirically to the product size, with the Effort E being commonly expressed in Person Months (PM) and the size in Kilo Lines of Code (KLOC) or in Kilo Delivered line source instruction KDSI or in Functions Points (FP) . Typical models used to express this relationship are of the form:

$$E = a\ Size^{b} + c \quad \text{---------------------------------} \ (1)$$

where a, b, and c in equation (1) are determined by regression analysis using a collection of project outcomes.

Table 1. Inverted form of LOC based formulae

Effort(E in PM) estimation models		
Basic COCOMO organic	EFFORT = 2.4 KLOC $^{1.05}$	KLOC = 0.43 EFFORT $^{0.952}$
Basic COCOMO semidetached	EFFORT = 3.0 KLOC $^{1.12}$	KLOC = 0.37 EFFORT $^{0.893}$

Basic COCOMO embedded	EFFORT = 3.6 KLOC $^{1.20}$	KLOC = 0.34 EFFORT $^{0.833}$
Intermediate COCOMO Organic	EFFORT = *3.2 KLOC* $^{1.05}$	*KLOC= 0.31 EFFORT* $^{0.952}$
Intermediate COCOMO semi detach	EFFORT = *3.0 KLOC* $^{1.12}$	*KLOC=0.33 EFFORT* $^{0.892}$
Intermediate COCOMO embedded	EFFORT = *2.8 KLOC* $^{1.20}$	*KLOC=0.35 EFFORT* $^{0.833}$
COCOMO II.2000	EFFORT = 2.9 KLOC $^{1.10}$	KLOC = 0.38 EFFORT $^{0.909}$
Halstead *Model*	EFFORT = 0.7 KLOC $^{1.50}$	KLOC = 1.27 EFFORT $^{0.667}$
Walston-Felix *Model*	EFFORT = 5.2 KLOC $^{0.91}$	KLOC = 0.16 EFFORT $^{1.10}$
Bailey-Basil *Model*	EFFORT = 5.5 KLOC $^{1.16}$	KLOC = 0.23 EFFORT $^{0.862}$
Doty (for KLOC > 9) *Model*	EFFORT = 5.288 KLOC $^{1.047}$	KLOC = 1.27 EFFORT $^{0.674}$

The different above model are used to calculate the estimated Effort. The estimation accuracy is improved by setting a number of parameters to reflect the situation at hand regarding the product, process, platform, and people involved as is done for example in COCOMO II [4].

From Table 1, we get the relation like

$$\text{Size} = a\,(\text{Effort})^{b} \text{ ----------------------- } (2)$$

3. FUNCTIONAL SIZES IN INCREMENTAL DEVELOPMENT

Suppose that Effort i.e. the total allocated project Effort, is fixed for a given development project and that the nature of the project is such that it can be broken into a sequence of n increments. The Effort E exerted in the individual increments is denoted by E1, E2, E3….En where

$$E= \sum_{i=1}^{n} Ei \text{ ---------------------- } (3)$$

and Duration D= c E ^{d} where c and d are constant values i.e.

$$D= c\,\left(\sum_{i=1}^{n} Ei\right)^{d}$$

To get the target functionality there is need of additional incremental development Effort used for the integration of the different increments. The term used for this is breakage as some of the existing code and design has to be mended to fit new increments. The breakage effect will be accounted by some fraction of person month. The incremental Effort exerted in increment i is gained as system enhancement. This fraction is termed as Additional Multiplier (AM). The net Effort is expressed as AM_i x E_i.

Kan asserted that 20% of the added code in staged and incremental releases of a product goes into changing the previous code [19]. In a recent paper the authors argued that the incremental integration breakage can be expected to lie in a range from 5% to 30%. This relates to the range in Additional Multipliers of 95% down to 70% in approximate terms [5]. Thus an Additional Multiplier (AM) can assume a range of values. The AM will equal to 1 when all of the gross work delivers new functionality where some gluing, integration or refactoring effort is needed. In this way the efficiency will be reduced below 1. The additional factor can exceed 1 in the case of marked code reuse (thus suggesting negative entropy and a major benefit from the application of reuse). One might ask if the Additional Multiplier (AM) can be negative in which case adding an increment is counterproductive i.e. more code needs to be mended than has been written. This interesting pathological case is outside the scope of this work. It is therefore assumed here that AM>0.

The net incremental effort exerted for increment i is {AM_i x E_i }. The Size of a developed increment is then derived from expression (2), we get

$$\text{Size}_i = a_i\,(AM_i \times E_i)^{bi} \text{ ---------------}(4)$$

This relation does not hold if the increments are "too small". For example, the COCOMO formulation stipulates that the size has to be larger than 2 KLOC (which roughly equates with 6 PM). Note also that for one-off development n=1 and AM=1 which is consistent with expression (2). When the outcome of the individual increments is accumulated the total developed Size say total size

$$\text{Total Size} = \sum_i Size_i \text{ ----------------------------}(5)$$

By combining 4th and 5th

$$\text{Total Size}= a_i \sum_i Size^{\,bi} \text{ -----------------------------}(6\,)$$

For homogeneous project and properties then a and b parameters in expression 6 are taken to constant values across the increments ie ai = a and b_i = b for all i=1,2……n . In this way the expression 6 will simplifies to

$$\text{Total Size} = a \sum_i (AMi * Ei)^{b} \text{ ------------------}(7)$$

Expression 6 and 7 are seen to represent a general way of estimating the outcome of incremental development projects. There is one question assisted "Does the number of increments affect the resulting product size? This question can be solved using expression 7 to get the answer.

First expression 2 is used to normalize expression 7 resulting in what is termed here is known as relative productivity say RP. As we know the productivity of software is defined as

Productivity= size / Effort

Then relative productivity RP= Total Size / (a E ^{b}). By combining (2) and (7)

$$RP=\sum_i (AMi * Ei\,/E)^{\,b} \text{ --------------------}(8)$$

which is the indication of productivity in incremental development . For RP > 1 productivity has to increased and if RP <1 productivity has to decreased.

Second let the Total Effort is equally divided into n increments then Ei=E/n for all i

$$RP = (1/n)^{b} \sum_i (AMi)^b \text{-----------------(9)}$$

Third a parameter termed as cumulative effort (CE) is defined as

$$CE = 1/n \sum_i (AMi)^b \text{------------------------(10)}$$

CE is function of n. If the case may be linear then b=1 then CE is treated as arithmetic average of AM_i. In general cumulative effort gain CE can be treated as a complex function of project. By combining expression 9 and 10 we get the new expression like

$$RP=(1/n)^{(b-1)} (1/n) \sum_i (AMi)^b$$

By again reducing it we get

$$RP = (n)^{(1-b)} CE \text{-------------------------------(11)}$$

Based on the COCOMO Basic, COCOMO – Intermediate, COCOMO -II, Bailey-Basil, Doty (for KLOC > 9), value of b is restricted between 0 and 1 (see Table 1) .In the case where cumulative effort CE is supposed to be constant then the expression 11 is seen to be a monotonic increasing function of n i.e. the productivity increases with increasing number of increments.

Further we know that

$$\text{duration} \propto \text{effort} \text{----------------------(12)}$$

For incremental developments we can write the equation (12) as

$$\text{duration} \propto \text{cumulative effort}$$

i.e. $D \propto CE$

hence we conclude that if the cumulative effort increases then duration will increases.

4. DISCUSSION

Table 2 shows computations of relative productivity RP using expression 11. Relative productivity RP for the exponent fixed at b=0.833. The value b=0.833 of *Intermediate COCOMO embedded* is chosen from Table1.

Table 2

	CE -Cumulative Effort			
n	1	0.9	0.8	0.7
1	1.00	0.90	0.80	0.70
2	1.12	1.01	0.90	0.79
3	1.20	1.08	0.96	0.84
4	1.26	1.13	1.01	0.88
5	1.30	1.17	1.04	0.91
6	1.35	1.21	1.08	0.94
7	1.38	1.24	1.10	0.96

It is clear that as the increments increases, the relative productivity increases. It is also interesting to find out what will be effect on relative productivity for a fix number of increments for constant value of b. Table 3 exemplifies the effect of varying b in a case where the number of increments is fixed say n=18.

Table 3

	CE Cumulative Effort			
b	1.0	0.9	0.8	0.7
0.833	***1.62***	***1.48***	***1.29***	***1.13***
0.892	***1.36***	***1.22***	***1.08***	***0.95***
0.952	***1.14***	***1.02***	***0.91***	***0.79***

It is found that the lowest value of b gain the highest value of relative productivity which is shown in bold font. If b=0.952, the line of Table 3 shows that RP<1 for the 0.8 and 0.7 values of CE. It is indicating that there is a loss in productivity whereas RP>1 for 1.0 and 0.9 values of CE indicating that there is high gain in productivity.

5. CONCLUSION

The paper investigates the implication of working on projects with fixed-effort level where increments may be utilized to obtain additional leverage for project managers .The productivity gain from incremental development in fixed effort situation comes when relative productivity RP>1 . The lowest values of b yield the highest gain in productivity .The authors are interested in addressing a number of emerging challenges that include like collecting empirical data on incremental development to validate the postulated formulation, analyzing the relationship between incremental delivery and team size, relating the findings to factors expressed in alternative estimation models, investigating the relationship between incremental development and the evolve ability of system. It is therefore hope that work put forward for further research and investigation.

6. REFERENCES

[1].G. R. Finnie, G. E. Witting, “A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models,” J. Systems Software vol. 39, pp. 281-289, 1997.

[2]. Martin Shepperd, Chris Schofield, “Estimating Software Project Effort Using Analogies,” *IEEE Transactions on Software Engineering*, vol. 23, no. 12, pp.736-743, 1997.

[3]. S. Chulani, B. Boehm, and B. Steece, “Bayesian analysis of empirical software engineering cost models,” IEEE Transaction on Software Engineerining, vol. 25, no. 4, July/August 1999.

T

[4]. Boehm, B. W., Abts, C., Brown, A. W., Chulani, S., Clark, B. K., Horowitz, E., Madachy, R., Reifer, D., and Steece, B. 2000. Software Cost Estimation with COCOMO II. P.H.

[5]. Benediktsson, O. and Dalcher, D. 2003 Effort Estimation in Incremental Software Development. IEE Proc. Softw., Vol. 150, no. 6, December 2003, pp. 351-357.

[6] M. Jorgensen, “A review of studies on expert estimation of software development effort,” Journal of Systems and Software, vol. 70, no. 1-2, pp. 37–60, 2004.

[7]. M. Jorgensen and K. Molokeen-Ostvoid, “Reasons for software effort estimation error: Impact of respondent error, information collection approach, and data analysis method,” IEEE Transactions on Software Engineering, vol. 30, no. 12, December 2004.

[8]. Pendharkar, P. C., Subramanian, G. H. and Rodger, J. A. 2005. A probabilistic model for predicting software development effort, IEEE Trans. Software Eng., 31, 7, 615-624.

[9]. Alaa F. Sheta, Estimation of the COCOMO Model Parameters Using Genetic Algorithms for NASA Software Projects, Journal of Computer Science 2 (2): 118-123, 2006

[10]. Mitat Uysal, Estimation of the Effort Component of the Software Projects Using Simulated Annealing Algorithm World Academy of Science, Engineering and Technology 41 2008.

[11]. Parvinder S. Sandhu, Porush Bassi, and Amanpreet Singh Brar Software Effort Estimation Using Soft Computing Techniques, World Academy of Science, Engineering and Technology 46 2008.

[12]. Parvinder S. Sandhu, Manisha Prashar, Pourush Bassi, and Atul Bisht , A Model for Estimation of Efforts in Development of Software Systems- World Academy of Science, Engineering and Technology 56 2009.

[13]. P. K. Suri1, Bharat Bhushan, Ashish Jolly, Time Estimation for Project Management Life Cycle: A Simulation Approach, International Journal of Computer Science and Network Security, VOL.9 No.5, May 2009.

[14]. Ch. Satyananda Reddy, Raju , A Concise Neural Network Model for Estimating Software Effort International Journal of Recent Trends in Engineering, Issue. 1, Vol. 1, May 2009.

[15]. Kirti Seth, Arun Sharma, Effort Estimation Techniques in Component Based Development - A Critical Review Proceedings of the 3rd National Conference; INDIACom-2009.

[16]. M. V. Deshpande ,S. G. Bhirud . Analysis of Combining Software Estimation Techniques International Journal of Computer Applications (0975 – 8887) Volume 5– No.3, August 2010.

[17]. Yogesh Singh, K.K.Aggarwal . *Software Engineering Third edition, New Age International* Publisher Limited New Delhi.

[18]. Pankaj Jolte , An Integrated Approach to Software Engineering Third edition Narosa Publishing house New Delhi.

[19]. Kan, S.H. 2003. Metrics and Models in Software Quality Engineering, 2nd edition Pearson Education.

[20]. Roger S Pressmen, "Software Engineering - A Practitioner's Approch" 6th Eddition Mc Graw Hill international Edition, Pearson education, ISBN 007 - 124083 – 7.