International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 23 Issue 9 – DECEMBER 2016.

A Comparative Analysis of Function Point Analysis for the Deployment of Software Projects

I. Priyanka[#1]

[#] **M.Phil. Scholar, Dept. of Computer Science, *S.vellaichammy Nagar college, Madurai, India***

***Abstract*— The appearance of the Function Point technique is practice of software measurement, with respect to the use of the traditional "Lines of Code approach". A FP count, however, requires a complete Functional Specifications of the software system, to be performed. There are two situations in which having an estimation method. The first case occurs when the development or enhancement project is in such an early phase that it is simply not possible to perform a FP count. an evaluation of the existing software asset is needed for required time and resources to perform a detailed FP calculation are not available. Based on these situations, the demand of methods for estimating - not counting - Function Points has risen from the organizations involved in software business. This paper presents, the estimation methods (Early Function Points, ILF Models, Backfiring) and a general benchmarking model, useful for the evaluation of any additional method, as well.**

***Index Terms*— software measurement, size, function point, estimation, benchmarking**

I. INTRODUCTION

FP Analysis is an efficient software sizing technique but, considering the standard IFPUG Counting Practices, it implies detailed set of descriptive documentation of the user functional requirements for any software application to be measured. There are two estimation method, compatible to the standard rules for FP, could be decisive. The first case occurs when the software development is in an early phase that it is simply not possible to perform a FP count.A standard count requires the identification of elements need for effort, time and cost forecasts based on size evaluation in Functional Specification. The second case occurs when an evaluation of software size asset is needed, but the required time and resources to perform FP calculation are not available. Various estimating methods have been proposed to respond to the need of evaluating the size of a software project.

II. BACKGROUND AND RELATED WORK

A. Background

Function Points is a cost estimating relationship (CER) to software costs. Once determined,Function Points are widely reported to be well suited for measuring the size of management information system (MIS), database intensive. Function Points are indirect quantitative measures of application software functionality and size. This procedure is composed of three logical divisions, finding the unadjusted function point count, value adjustment factor, and Function Points. Finding the unadjusted function point count consists of counting the number of external inputs, external outputs, external inquiries, internal logical files, and external interface files. Finding the value adjustment factor consists of rating system, input and output, and application complexity. Finding Function Points consists of factoring unadjusted function points and value adjustment factor together.

Function Points have two distinct purposes. The first one is basis for software measurement, comparison, and analysis. The second is to determine software size for input into software cost estimation models are based on empirical cost estimating relationships (CERs) between Function Points and effort.

B. Related Work

Estimation by expert [18][19], analogy based estimation schemes [20], algorithmic methods including empirical methods [21], rule induction methods [22], artificial neural network based approaches [23] [24] [25], Bayesian network approaches [26], decision tree based methods [27] and fuzzy logic based estimation schemes [28][29]. The estimation parameters are commonly derived from empirical data. Approximate effort and cost estimation of software applications continues to be a critical issue for software project managers [32]. Expert judgment remains widely used in applying statistics and machine learning techniques to predict software project effort [33][34]. Hardware costs, travel and training costs and effort costs are the three principal components of cost[36][37]. Many research papers is able to provide accurate effort/cost estimation is still a challenge for many reasons. They include: (i) the uncertainty in collected measurement, (ii) the estimation methods have many drawbacks and (iii) the cost drivers is along with the development environment which might not be clearly specified [38]. The most popular algorithmic estimation models include Boehm"s constructive cost model (COCOMO) [39]. Thus, accurate estimation methods, for example, the FP method, have gained increasing importance [40]. . The size is determined by identifying the components

of the system as seen [41] by the end-user: the inputs, outputs, inquiries, interfaces [42] to other systems and logical internal files [43]. The components are classified as simple, average or complex. All values are then scored and the total is expressed in unadjusted FPs (UFPs) performance and complexity of processing can be used to weigh the UFP. The result of these computations is correlates to system size. The FP metric does not correspond to any actual physical attribute of a software system [47,48] (such as lines of code or the number of subroutines) is useful for measuring productivity, and estimating the amount a development effort and time needed for a project [49,50]. The total number of FPs depends on processing logic types in the following five classes [51]. It is well documented that the software industry suffers from frequent cost overruns [52]. A contributing factor is, we believe, the imprecise estimation terminology in use. A lack of clarity and precision [53] in the use of estimation terms reduces the interpretability of estimation [54] accuracy results, makes the communication of estimates difficult and lowers the learning possibilities [55]. Number of enhancements to adjustment factors is introduced. This model is grouping into three groups. They are "System complexity", "I/O complexity" and "Application complexity".

III. COMPARATIVE OF ESTIMATION METHODS

Effort (and size) estimations are critical to the success of a software project [8]. Of the different techniques presented in prior literature [9], the most popular include algorithmic and parametric models, expert judgment, and reasoning by analogy [10]. According to Heemstra's survey, 29 different software-based cost models have been proposed since 1966 [11].These models usually take a software size as input to estimate development effort. The LOC-based models are mostly nonlinear, and their estimated effort is at least quadratic to the size. The generation process of counting function points involves nonlinear computation. Machine learning models for size and/or effort estimation are rather more recent; they include case-based reasoning [13], fuzzy logic [14], neural network [15], and many others [16]. Most such approaches report results that are comparable to those of other techniques [17]. However, because the data needed by these models most can be applied only during the later stages of the software development process. The most common estimation methods are,

A. *Direct Estimation Methods*

Direct Estimation Methods approach to estimation may results in very accurate estimates, although it is entirely dependent on the experience of the expert(s). Sometimes, in case of many experts collaborating to the same effort, it may be difficult for the estimates to converge to a unique value. The estimate itself may be influenced by subjective factors, as personal relationships, contractual aspects, and so on.

B. *Delphi Or Shang Techniques*

The most commonly used procedures, by which individual predictions are combined. In shortly, they provide iterated cycle of anonymous estimations by each expert, until the estimates converge to an acceptable range. The conclusion is a group estimate arrived at by consensus. The group estimate is typically a better overall estimate than any individual prediction.

C. *Three-Point Estimation Techniques*

Three-Point Estimation Technique is

to improve direct estimation, when more values are provided by estimators. Given the Minimum, the Most Likely, and the Maximum Value for the size,

the estimate is:

Est.Size = (Min + 4×MostLikely + Max) / 6

with standard deviation:

s= (Max - Min) / 6

D. *Simple Analogy Method*

Simple Analogy Method is in the historical database, that is "similar" to the application under estimation. The found implemented system provides a quick estimate of the new project size. Further investigation likely leads to Structured Analogy

E. *Structured Analogy Methods*

The estimator compares the proposed application to one or more existing applications: This type of application, establish an initial prediction and passing from a Simple Analogy to a Structured Analogy, the differences and similarities are identified and used explicitly in a mathematical way to adjust the estimate. A concept of "distance" among systems may be defined and used to prioritize the choices.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

A. *Data Sets*

Sr. No	System	FP		UCP	
		System Size (Count)	Effort (Person-months)	System Size (Count)	Effort (Person-months)
1	A	53.68	6.31	63.08	7.01
2	B	59.92	7.27	66.65	7.41
3	C	81	10.73	101.1	11.23
4	D	68.32	8.61	79.7	8.86
5	E	78.84	10.36	98.30	10.92
6	F	49.02	5.61	57.90	6.43
7	G	58.86	7.11	64.59	7.18
8	H	80.64	10.67	101.10	11.23
9	I	53.53	6.29	61.05	6.78
10	J	69.58	8.82	92.03	10.23

TABLE.1. DEVELOPMENT EFFORT

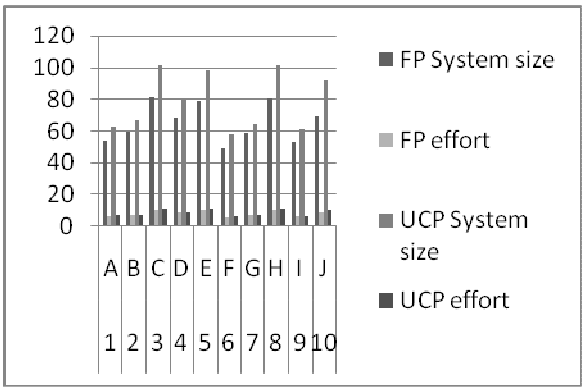

Figure 1: chart

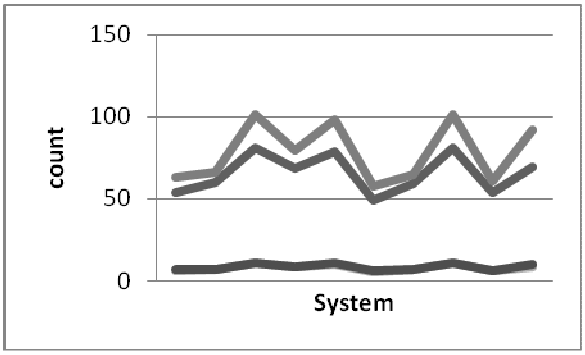

Figure 2: graph

V. CONCLUSION

A number of different models and effort estimation methods have been developed in the past four decades. This clearly indicates the awareness among the researchers of the need to improve effort estimation in software engineering. Many factors have impact on the software development process. These factors are not only human, technical but also political and their impact can never be fully predicted. The even insufficiently accurate estimates are far better than none. We have illustrated result of two approaches for measuring the size in the estimation process in our work. If the estimation is done accurately, it decreases error. We have analyzed the values of different matrix under function point as well as use case point method. The relationship among the matrix of these methods shows the practical results. Hence we conclude that the estimation process through use case point is better than the function point. The requirement & design is more clearly shown in use case point model than in function point model. The estimation using use case point shows practical reality of development. Hence the accuracy of estimation depends upon method used for estimation also. Although effort estimation is a critical step for the success of a software project, in practice, many projects still use ad hoc methods to conduct this task. This phenomenon may result because most of the generic algorithmic estimation models are difficult to be adopted, in that they require the collection of many detailed items that may affect the size (and effort) associated with an application. This research presents a different approach that trades "generic" with "specific" and greatly simplifies the estimation model to enable common programmers to use it. Although the new approach may not be as formal as a normal estimation model, it does reduce the difficulties of using it by giving programmers a method to build their own model, which captures more of a given application domain's characteristics and is easier to use. The demonstration of this approach offers an unique benefit in that it makes the function classification more suitable for a particular application domain so that function point counting can be conducted by programmers themselves instead of by a certified FPA expert. We believe such an approach should be applicable to domains when FPA works since it still stays with the FPA spirit. However, it may shares the limitations of FPA also. To verify its usefulness, it is no doubt that more experimental data from a wider range of application domains and environments should be collected to deepen the investigation into the trade-off between generic versus specific. Atleast two research directions exist. First, studies could evaluate whether the tailored FPA model always fits a business application domain and still maintains comparable estimation accuracy. Second, the ideas presented here in might be adopted to another estimation model and determine how it performs. Each direction will lead to more detailed research findings..

REFERENCES

[1] AD/M Productivity Measurement and Estimate Validation; Allan Albrecht, IBM Corporate Information Systems and Administration; 1985
[2] Function Point Analysis: Difficulties and Improvements, C. R. Symons, IEEE Transactions on Software Engineering, 1985.
[3] Software Function, Source Lines of Code and Development Effort Prediction: A Software Science Validation; Albrecht A.J., & Gaffney J.E.; IEEE Transactions on Software Engineering, Vol SE -9, No. 6; November 1983.
[4] Empirical Studies of the Assumptions Underlying Software Cost Estimation Models; B. A. Kitchenham, NCC Ltd; 1991.
[5] Function Point Counting Practices Manual: Release 3.0 ; IFPUG Counting Practices Committee; 1990.
[6] Software Sizing and Estimating: Mk II FPA; Symons C.R.; John Wiley & Sons; 1991
[7] A Cooperative Industry Study: Software Development/Maintenance Productivity; Xerox Corporation;1985
[8] M. Jorgensen and D. I. K. Sjoberg, "Impactof effort estimates on software project work," Information and Software Technology, Vol. 43, 2001, pp. 939-948.
[9] C. Gencel and O. Demirors, "Functional size measurement revisited," ACM Transactions on Software Engineering and Methodology, Vol. 17, 2008, pp. 1-36.
[10] B. Barry, C. Abts, and S. Chulani, "Software development cost estimation approaches −A survey," Annals of Software Engineering, Vol. 10, 2000, pp. 177-205.
[11] A. L. Lederer and J. Prasad, "A causal model for software cost estimating error," IEEE Transactions on Software Engineering, Vol. 24, 1998, pp. 137-148.
[12] A. J. Albrecht and J. E. Gaffney Jr., "Software function, source lines of code, and development effort prediction: A software science validation," IEEE Transactions on Software Engineering, Vol. 9, 1983, pp. 639-648.
[13] R. Bisio and F. Malabocchia, "Cost estimation of software projects through case-base reasoning," in Proceedings of Case-Based Reasoning Research and Development, 1995, pp. 11-22.
[14] O. D. Lima, P. M. Farias, and A. D. Belchior "Fuzzy modeling for function points analysis, "Software Quality Journal, Vol. 11, 2003, pp. 149-166.
[15] J. Hakkarainen, P. Laamamen, and R. Rask, "Neural networks in specification level software size estimation," in P. K. Simpson, ed., Neural Network Applications, IEEE Technology Update Series, 1993, pp. 887-895.
[16] A. Heiat, "Comparison of artificial neural network and regression models for estimating software development effort," Information and Software Technology, Vol. 44, 2002, pp. 911-922.

[17] A. R. Gray and S. G. MacDonell, "A comparison of techniques for developing predictive models of software metrics," Information and Software Technology, Vol. 39, 1997, pp. 425-437.
[18] SaleemBasha, Dhavachelvan.P. "Analysis of Empirical Software Effort Estimation Models" (IJCSIS) International Journal of Computer Science and Information Security, Vol. 7, No. 3, 2010.
[19] Jorgen MS joberg D.I.K, "The Impact of Customer Expectation on Software Development Effort Estimates" International Journal of Project Management, Elsevier, pp 317-325, 2004.
[20] Chiu NH, Huang SJ, "The Adjusted Analogy-Based Software Effort Estimation Based on Similarity Distances," Journal of Systems and Software, Volume 80, Issue 4, pp 628-640, 2007.
[21] Kaczmarek J, Kucharski M, "Size and Effort Estimation for Applications Written in Java," Journal of Information and Software Technology, Volume 46, Issue 9, pp 589-60, 2004.
[22] Jeffery R, RuheM,Wieczorek I, "Using Public Domain Metrics to Estimate Software Development Effort," In Proceedings of the 7th International Symposium on Software Metrics, IEEE Computer Society, Washington, DC, pp 16–27, 2001.
[23] Heiat A, "Comparison of Artificial Neural Network and Regression Models for Estimating Software Development Effort," Journal of Information and Software Technology, Volume 44, Issue 15, pp 911-922, 2002.
[24] K. Srinivasan and D. Fisher, "Machine learning approaches to estimating software development effort," IEEE Transactions on Software Engineering, vol. 21, pp. 126-137, 1995.
[25] A. R. Venkatachalam, "Software Cost Estimation Using Artificial Neural Networks," Presented at 1993 International Joint Conference on Neural Networks, Nagoya, Japan, 1993.
[26] G. H. Subramanian, P. C. Pendharkar, and M. Wallace, "An Empirical Study of the Effect of Complexity, Platform, and Program Type on Software Development Effort of Business Applications, "Empirical Software Engineering, vol. 11, pp. 541-553, 2006.
[27] R. W. Selby and A. A. Porter, "Learning from examples: generation and evaluation of decision trees for software resource analysis," IEEE Transactions on Software Engineering, vol. 14, pp. 1743-1757, 1988.
[28] S. Kumar, B. A. Krishna, and P. S. Satsangi, "Fuzzy systems and neural networks in software engineering project management,"Journal of Applied Intelligence, vol. 4, pp. 31-52, 1994.
[29] Huang SJ, Lin CY, Chiu NH, "Fuzzy Decision Tree Approach for Embedding Risk Assessment Information into Software Cost Estimation Model," Journal of Information Science and Engineering,Volume 22, Number 2, pp 297–313, 2006.
[30] M. van Genuchten and H. Koolen, "On the Use of Software Cost Models," Information & Management, vol. 21, pp. 37-44, 1991.
[31] T. K. Abdel-Hamid, "Adapting, Correcting, and Perfecting softwareestimates: Amaintenance metaphor " in Computer, vol. 26, pp. 20-29, 1993
[32] K. Maxwell, L. Van Wassenhove, and S. Dutta, "Performance Evaluation of General and Company Specific Models in Software Development Effort Estimation," Management Science, vol. 45, pp. 787-803, 1999.
[33] H. Azath,and R.S.D. Wahidabanu "Efficient effort estimation system viz. function pointsand quality assurance coverage" IETSoftw., 2012, Vol. 6, Iss. 4, pp. 335–341 335, doi: 10.1049/iet-sen.2011.0146.
[34] Deng, J.D., Purvis, M.K., Purvis, M.A.: „Software effort estimation: harmonizing algorithms and domain knowledge in an integrated data mining approach", Inf. Sci. Discuss. Pap. Ser., 2009, 2009, (5), pp. 1 –13
[35] Idri, A., Khoshgoftaar, T.M., Abran, A.: „Can neural networks be easilyinterpreted in software cost estimation?". 2002 World Congress on Computational Intelligence, Honolulu, Huwaii, 12–17May 2002, pp. 1–
[36] Mittal, H., Bhatia, P.: „A comparative study of conventional effort estimation and fuzzy effort estimation based on triangular fuzzy numbers", Int. J. Comput. Sci. Secur., 2002, 1, (4), pp. 36–47.
[37] Benton, A., Bradly, M.: „The International Function Point User Group(IFPUG)", in „Function point counting practices manual –release 4.1"(SA, 1999).
[38] Aljahdali, S., Sheta,A.F.: „Software effort estimation by tuning COOCMOmodel parameters using differential evolution". Int. Conf. on Computer Systems and Applications (AICCSA), 16–19 May 2010, pp.1-6.
[39] Boehm, B.W.: „Software engineering economics" (Prentice Hall,Englewood Cliffs, NJ, 1981).
[40] Peischl, B., Nica, M., Zanker, M., Schmid, W.: „Recommending effort estimation methods for software project management". Proc. IEEE/WIC/ACM Int. Conf. on Web Intelligence and Intelligent Agent Technology, Milano, Italy, 2009, vol. 3, pp. 77–80
[41] B.W. Boehm, "Software Engineering Economics," Prentice Hall, 1981.
[42] B.W. Boehm, E. Horowitz, R. Madachy, D. Reifer, B. K. Clark, B.Steece, A. W. Brown, S. Chulani, and C. Abts, "Software CostEstimation with COCOMO II," Prentice Hall, 2000.
[43] F. J. Heemstra, "Software cost estimation," Information and Software Technology, vol. 34, pp. 627-639, 1992
[44] N. Fenton, "Software Measurement: A necessary Scientific Basis,"IEEE Transactions on Software Engineering, vol. 20, pp. 199-206, 1994.
[45] Barry Boehma, Chris Abtsa andSunitaChulani, "Softwaredevelopment cost estimation approaches -A survey" Annals of Software Engineering, pp 177-205, 2000.
[46] James Nelson, H., Monarchi, D.E.: „Ensuring the quality of conceptual representations", Softw. Qual. J., 1997, 15, (2), pp. 213–233.
[47] Khoshgoftaar, T.M., Allen, E.b., Naik, A., Jones, W.D., Hudepohl, J.P.:„Using classification trees for software quality models: lessons learned",Int. J. Softw. Eng. Knowl. Eng., 1999, 9, (2), pp. 217–231
[48] Kitchenham, B.A.: „Cross versus within-company cost estimation studies:a systematic review", IEEE Trans. Softw. Eng., 2007, 33, (5), pp. 316–329
[49] Hannay, J.E., Sjøberg, D.I.K., Dyba, T.: „A systematic review of theory use in software engineering experiments", Softw. – Pract. Exper., 2007,33, (2), pp. 87–107
[50] Jack E. Matson, Bruce E. Barrett, and Joseph M. Mellichamp, "Software Development Cost Estimation Using Function Points" IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 20, NO. 4, APRIL 1994
[51] CHRIS F. KEMERER "An Empirical Validation of Software Cost Estimation Models"
[52] Putnam. L.H. General empirical solution to the macro software sizing estimating problem. IEEE Trans. Soffw. Eng. SE 4, 4 (July 1978), 345-361.
[53] Putnam. L.. and Fitzsimmons, A. Estimating software costs. Datamation 25, lo-12 (Sept.-Nov. 1979)
[54] B Boehm, C Abts, and S Chulani. "Software Development Cost Estimation Approaches – A Survey", Technical Report USC-CSE-2000-505", University of Southern California – Center for Software Engineering, USA, (2000).
[55] S. chulani, B. Boehm, and B. Steece, "Bayesian Analysis of Emperical Software Engineering Cost Models," IEEE Trans. Software Eng., vol.25, no. 4, pp.573-583, 1999.
[56] M.Pauline, P.Aruna and B.Shadaksharappa "Fuzzy-Base Approach Using Enhanced Function Point to Evaluate the Performance of Software Project", The IUP Journal of Computer Sciences, Vol. VI, No. 2, 2012