ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 11(48), DOI: 10.17485/ijst/2018/v11i48/138343, December 2018* ISSN (Online) : 0974-5645

Software Effort Estimation using Function Point based Clustering Technique (FPBCT)

N.A. Bhaskaran* **and V. Jayaraj**

School of Computer Science Engineering and Applications, Bharathidasan University, Tiruchirappalli - 620 024, Tamil Nadu, India; nabhaskaran@gmail.com, jaya_v2000@yahoo.com

Abstract

Objectives: To estimate software development effort efficiently and easily. **Methods/Statistical Analysis**: Estimations at the start of the project can help identify functions involved and connect with entire the effort and implementation of a project. The proposed technique called FPBCT (Function Point Based Clustering Technique) is based on function point analysis and k-means clustering technique for achieving better predictive accuracy in software effort estimation. A live case of software development in catering is taken as a sample for study. **Findings:** The food service industries provide catering services for various occasions such as School functions, Colleges functions, Hospital functions, Marriage Party and many other formats, including 'on-premises' and 'off-premises' catering services. FPBCT groups the input values into three clusters based on their Euclidean distance measures of three centroids namely 3 for elementary, 5 for Medium and 8 for complex functions in a software program. In each centroid, the clusters are formed based on similarity measures i.e. time and lines of code and thus FPBCT effectively estimates the effort required to program software programs or modules based on function points. This research takes into account the function points in software, groups them and estimates the effort required for a project. Further, it also gives comparative efforts required in different programming languages for estimating efforts required using software function points. **Application/Improvements:** The proposed technique can also be adapted to object oriented programming using use cases in the future.

Keywords: Clustering, k-Means, Lines of Codes, Software Development, Software Effort Estimation, FBCT

1. Introduction

Software development is a creative process where personal efficiencies differ, making it difficult to plan and estimate. Software development effort estimation is the foresight on effective utilization of resource efforts required in programming. It is necessary for organizations to complete projects within a stipulated budget and time. The success of projects directly depends upon effort and cost estimations. Effort estimation becomes an important parameter as cost is based on the total effort required for completing a project. Software development cost estimation is a future prediction of a work that is managed by managers who decide on the time frames and resources required for project completion.[1] It is important for both customers and software companies to plan and predict parameters in a project at the initial stage to maximize outputs. Thus, Project Managers determine the cost and effort for allocating the right budgets. The evaluation of effort and cost is achieved using project metrics of previous projects for comparisons, before arriving at a conclusion on the time frame required for new projects. Thus, software estimation becomes a crucial activity for successful software project management and fundamental to the project planning and budgeting.

Software Effort Estimation (SEE) can be categorized into Algorithmic and non-algorithmic models. Non-Algorithmic methods estimate by analyzing previous project information datasets. They do not use any formula or statistical techniques for effort estimations. They stick to comparisons between two sets of information for their conclusions or results. On the other hand, Algorithmic models (Parametric models) use mathematical formulae on inputs for their estimations. Most commonly used algorithmic models for software cost estimation include Boehm's COCOMO,[2] Albrecht's Function Point Analysis,[3] and Putnam's SLIM.[4] The primary focuses of the algorithmic model is design, analysis, implementation and

*Author for correspondence

optimization. The secondary focus is on the experimental evaluation and practical applications of the software. Though, a significant improvement has been achieved using soft computing techniques in SEE, limitations do exist. The vague nature of software requirements also complicates SEE. It is unlikely to expect very accurate SEE because of the inherent uncertainty in software development projects and the complex and dynamic interaction of factors that impact software development.

The projects may have deficit or vague of information which need extra efforts to acquire or understand. The risk in a project rises when important functions are identified at the end of the project impacting effort estimations and increasing its life cycle. The quality of the estimation is another factor, which determines the success of the project and facilitates in avoiding risks. On-time deliverables are an important concern for software organizations and inaccurate estimations have frequently leads to project failures. Under-estimated projects lead to unfavorable outcomes on business. While over-estimated projects result in the poor resource allocations and missed business opportunities. Further, estimation data is inherently non-linear, making accurate estimations difficult. Better estimations can be achieved, when this non linearity is eliminated using relationships and reducing heterogeneity by grouping. The main objective of this research is to classify software functions based on complexity, group them and add weights to the factors for SEE.

2. Effort Estimation

Estimation is an approximated value on input data which may be incomplete or uncertain. It determines the effort, money, time and resources required to build a specific system or a product. Estimations are based on previously available and recorded data about assumptions and identified risks. The basic steps in project estimation involve estimating the size, effort, schedule and cost involved in the project. Project metrics provide valuable inputs for generating quantitative estimates. Generically, Project Estimation approaches use decomposition Techniques, where the complete project is divided into smaller manageable tasks and estimated. The effort required for a project can be estimated by breaking down a project into related software engineering activities. For example, identifying and dividing these activities into tasks that can be measured and estimate the effort hours or days or months per person. The total of all efforts is the cumulative effort required for the project. Function Point (FP), introduced by Alan Albrecht of IBM in 1979, is a unit for expressing business functionality quantitatively. A study compared the accuracy of the four estimation models in software projects using FPA for effort estimations.[5] FPs is widely accepted as an industry standard for functional sizing and help measure the software size. Several recognized standards for FPs are in existence like ISO Standards (COSMIC – ISO/IEC 19761:2011 Software engineering). Function Point Analysis (FPA) quantifies the functions contained within software in terms that are meaningful to the software users and are based on requirements specification. A case study applying both the COCOMO model and Function Point Analysis for SEE was done showing the estimated effort and error percentage.[6] Elementary Process in FP is the smallest functional user requirement unit that constitutes a complete transaction and is self-contained. There are two types of functions namely data and transaction functions. An Application boundary and FP elements are depicted in Figure 1.

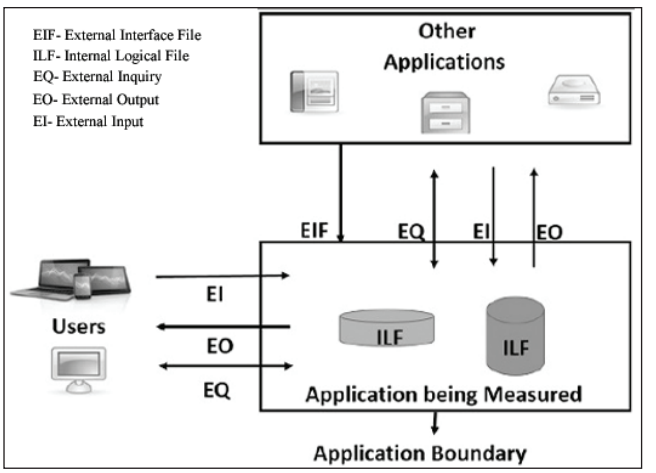

Figure 1. FP Elements in the Application Boundary.

SEE are crucial for organizations as it provides expense management. Though many quantitative models in estimation have been proposed an overwhelming majority of these models consider Line of Code (LOC) and FPs for their estimations. Project size estimations affect the accuracy of the expense estimations.[7]

3. Clustering and Estimation

A number of data clustering techniques have been developed to find the optimal subsets of data from the existing

datasets.[8-10] Clustering is an unsupervised classification; partitions unlabeled sets of data into smaller sets based on similarity measures. Clustering can be useful to experts in gathering related software product modules. Such a system influences after grouping strategies to gathering comparable modules together as intelligible clusters, and facilitates the tedious issue of marking for the expert.[11] Every cluster can represent a module for further review by an expert in the marked groups. The similarity measures used for grouping can also ease the experts work in adding further programs to modules. Once a program or module is identified, clustering can help in an abstract effort estimation of the programming tasks and effectively modular estimations. Clustering can be used as a preprocessing step to break down complicated software projects into smaller groups and hence tasks for effort estimations. Error-prone parts can also be identified and such information can be reduced.[12] K-means clustering algorithm is a simple and efficient algorithm to implement for categorizing data into clusters. Its main objective is to find cluster centers or centroid which is done by minimizing the distance between clusters. The total number of clusters is pre-defined in advance. K-means approach[13-15] can be used with two variants/criteria's in order to measure the coherence of clusters. K-means clustering is depicted in equation (1).

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} || x^{(j)} - c_j ||^2 \qquad (1)$$

Where J is the objective function, k-cluster numbers, x is a case, n is number of cases and C the centroid.

4. Proposed Function Point based Clustering Technique (FPBCT)

SEE can be predicted by machine learning techniques due to its nature of learning. These techniques focus on data analysis. Researchers using different techniques are more concerned about accurate predictions in effort estimations while developing software. Estimations at the start of project can help identify functions involved in learning resource difficulties from the project. This kind of earlier estimation connected with entire effort and implementation is extremely important because this specific estimation (both effort as well as time) is needed for overall improvements in project developments. FPBCT takes into account the atom of effort estimations by considering functional points in its analysis and applies k-means clustering for its effort estimation. The FPBCT algorithm is listed below:

Input: Functional Point Module Dataset with X data points FP = {FP_1, ...,FP_x}
Where FP is the Function Points and X is finite
And P = {M_1….M_N}
Where M-Modules and N is finite

Step 1: For Each Module do
For I = 1 to N do
If FP_I= 'C' then
Multiply Complexweight to FP_I
Else If FP_I= 'M' then
Multiply Medium weight to FP_I
Else If FP_I= 'E' then
Multiply Elementary weight to FP_I
End if
End Function List
End Module

Step 2: K-Means Input: FP dataset with X data points FP = {FP_1, ...,FP_x} $FP_X \in$ {1, ..., X}.

Step 3: Declare the cluster centroid points for Complex, Medium and Elementary.

Step 4: Centroid prediction: For cluster c, Let M_N = {$FP_N|M_N$ = c}, the centroids predicted as μM_c = 1 $|FP_c|$ such that FP∈M.

Step 5: Stop if M does not change, otherwise go back to Step 3.

Similar cluster numbers form centroids. In each centroid the clusters are formed based on similarity measures i.e. time and lines of code.

5. Results

Catering is multifaceted as they may provide food service at a remote site or a site such as hotels, and hence there is a need of software application for maintaining master and transaction records and making system automated. The system has to support all activities related to the Menu Creation including master tables maintenance and report generation. The Master Maintenance module consists of information about the products and services including sub-modules, Items master (information about the particular Items), Team master (Expert team in making food items), Sub items master (detail to item master), Event master (Information on booked events), and Customer master (Information on Customers). A food menu module caters to customer requirements on events. The reporting

Modules output required reports from the software. The Food Service software top level functions are listed in Table 1. Table 2 lists the top level function as functions points in terms of complexity, while Table 3 details lines of code metrics of computer languages based on Software Economics and Function Point Metrics, Version 10.0 April 14, 2017.

Table 1. Food Service Software Top Level Functions

No	Function Description	Function Type
1	Customer Master	Add, Modify, Delete, View
2.	Item Master	Add, Modify, Delete, View
3	Item Sub-Master	Add, Modify, Delete, View
4	Team Master	Add, Modify, Delete, View
5	Event Master	Add, Modify, Delete, View
6	Customer Events Master	Add, Modify, Delete, View
7	Menu Creations	Add, Modify, Delete, View
8	Date wise Menus Display	Report
9	Display Menu handling Status Date Wise	Report
10	Display Team Task (Individual)	Report
11	Display Team Task (Customer/Party wise)	Report

Table 2. Food Service Software Function Points with Complexity

No.	Function Description	Function Point	Complexity
1	Customer Master	Add	C
2.	Item Master	Add	C
3	Item Sub-Master	Add	C
4	Team Master	Add	C
5	Event Master	Add	C
6	Customer Events Master	Add	C
7	Menu Creations	Add	C
8	Customer Master	Modify	C
9	Item Master	Modify	C
10	Item Sub-Master	Modify	C
11	Team Master	Modify	C
12	Event Master	Modify	C
13	Customer Events Master	Modify	C
14	Menu Creations	Modify	C
15	Customer Master	Delete	E
16	Item Master	Delete	E
17	Item Sub-Master	Delete	E
18	Team Master	Delete	E
19	Event Master	Delete	E
20	Customer Events Master	Delete	E
21	Menu Creations	Delete	E
22	Customer Master	View	M
23	Item Master	View	M
24	Item Sub-Master	View	M
25	Team Master	View	M
26	Event Master	View	M
27	Customer Events Master	View	M
28	Menu Creations	View	M
29	Date wise Menus Display	Report	C
30	Display Menu handling Status Date Wise	Report	C
31	Display Team Task (Individual)	Report	C
32	Display Team Task (Customer/Party wise)	Report	C

Table 3. Language-Complexity and lines of code metrics

Complexity	Language	Kilo Lines of Code	Work Hours / KLOC	Estimated Time / KLOC
Complex	Java	53.33	238.07	4.46409151
Complex	C	128	205.26	1.60359375
Complex	HTML	160	200.57	1.2535625
Medium	Java	53.33	258.07	4.83911494
Medium	C	128	235.26	1.83796875
Complex	HTML	160	230.57	1.4410625
Elementary	Java	53.33	268.07	5.02662666
Elementary	C	128	240.26	1.87703125
Elementary	HTML	160	245.57	1.5348125

FPBCT is tested on a dataset generated from a Catering Services Software requirement. The food service industries provide catering services for various occasions such as School functions, Colleges functions, Hospital functions, Marriage Party and many other formats, including 'on-premises' and 'off-premises' catering services. Table 4 lists the input table with complexities.

Table 4. FPBCT Input Table

No	Function Description	Function Point	Complexity / Weight
1	Customer Master	Add	8
2.	Item Master	Add	8
3	Item Sub-Master	Add	8
4	Team Master	Add	8
5	Event Master	Add	8
6	Customer Events Master	Add	8
7	Menu Creations	Add	8
8	Customer Master	Modify	8
9	Item Master	Modify	8
10	Item Sub-Master	Modify	8
11	Team Master	Modify	8
12	Event Master	Modify	8
13	Customer Events Master	Modify	8
14	Menu Creations	Modify	8
15	Customer Master	Delete	3
16	Item Master	Delete	3
17	Item Sub-Master	Delete	3
18	Team Master	Delete	3
19	Event Master	Delete	3
20	Customer Events Master	Delete	3
21	Menu Creations	Delete	3
22	Customer Master	View	5
23	Item Master	View	5
24	Item Sub-Master	View	5
25	Team Master	View	5
26	Event Master	View	5
27	Customer Events Master	View	5
28	Menu Creations	View	5
29	Date wise Menus Display	Report	8
30	Display Menu handling Status Date Wise	Report	8
31	Display Team Task (Individual)	Report	8
32	Display Team Task (Customer/Party wise)	Report	8

FPBCT the groups the input table into three clustersbased on their Euclidean distance measures of three centroids namely 3 for elementary, 5 for Medium and 8 for complex. The grouping of the input table in FPBCT based on the ***centroid***and elements in the group are listed in Table 5.

Table 5. FPBCT Clustering Output 1

	Centroid	Elements	Cluster Elements
Group 1	8	1-14, 29-32	19
Group 2	5	22-27	6
Group 3	3	15-21	7

FPBCT then takes the values into consideration the complexity and line metrics values of Table 3 for applying to the clustered output of Table 5 to arrive the effective efforts required in three languages namely C, Java and HTML for the Food Service Software and is listed in Table 6 and Figure 2.

Table 6. FPBCT Comparative KLOC/Hour of Languages

	Effective KLOC / Hr	Effective KLOC / Hr	Effective KLOC / Hr	Total Man Hours
Language→	Java	C	HTML	
Complex	0.558011	0.200449	0.156695	758.29679
Medium	0.967823	0.367594	0.288213	81.611834
Elementary	1.675542	0.625677	0.511604	33.56364

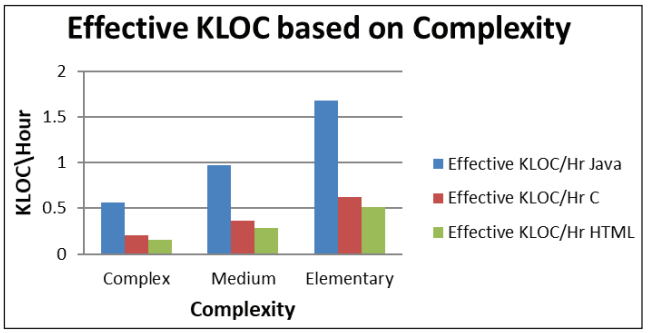

Figure 2. Effective KLOC of Languages based on Code Complexity.

It is evident from Figure 2 that less effort is required in programming in Java for three levels of programming when compared to C or HTML as more kilo lines of code can be duplicated in java when compared to C or HTML. Table 7 lists the final output table of FPBCT and depicted as Figure 3, where FPBCT implies the total man hour efforts estimated for three programming languages.

It is evident from Figure 3 that FPBCT can effectively estimate the effort required to program software programs or modules based on function points as identified by experts in software development.

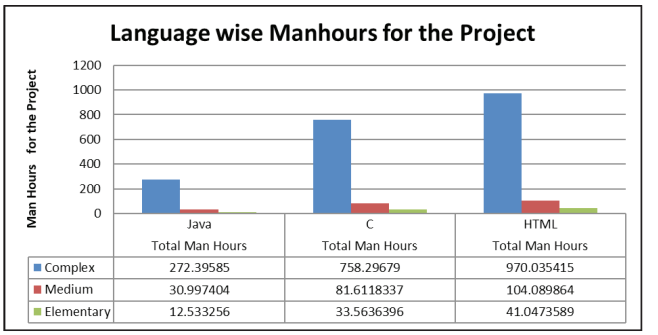

Figure 3. Language wise Estimated Effort for Food Service Software by FPBCT.

Table 7. FPBCT Effort Table

Clusters	Centroid	Elements	Cluster Element Count	Effective KLOC	Effective KLOC/ Hr Java (Table 3)	Total Man Hours Java	Effective KLOC/ Hr C (Table 3)	Total Man Hours C	Effective KLOC/ Hr C HTML	Total Man Hours HTML
Complex	8	1-14, 29-32	19	152	0.558011	272.39585	0.200449	758.29679	0.156695	970.035415
Medium	5	22-27	6	30	0.967823	30.997404	0.367594	81.6118337	0.288213	104.089864
Elementary	3	15-21	7	21	1.675542	12.533256	0.625677	33.5636396	0.511604	41.0473589

6. Conclusion

Software effort estimation (SEE) is a tedious task and project managers face problem in selecting a suitable effort based cost estimation techniques for calculating efforts and effectively the project cost. This research takes into account the function points in software, groups them and estimates the effort required for a project. Further, it also gives comparative efforts required in different programming languages for estimating efforts required using software function points. Challenges faced in effort estimation of software can be overcome with the proposed SEE using Function Point Based Clustering Technique (FPBCT). It can be concluded that FPBCT is a simple, effective technique based on clustering function points in software development projects and can be implemented to estimate efforts irrespective of the programming language chosen for development. It can help experts define a path for software development in cases where the number of complex modules or programs is too many and becomes tedious to arrive even at abstract effort estimation for development of software.

7. References

1. Khoshgoftaar TM, Seliya N. Analogy-based practical classification rules for software quality estimation. Empirical Software Engineering. 2003; 8(4):325-50. https://doi.org/10.1023/A:1025316301168
2. Boehm BW. Software Engineering Economics. NJ, Prentice Hall. p. 1-767.
3. Albrecht AJ, Gaffney JR. Software measurement, source lines of code, and development effort prediction: a software science validation. IEEE Transactions on Software Engineering. 1983; 9(6):639-48. https://doi.org/10.1109/TSE.1983.235271

4. Putnam LH. A General Empirical Solution to the Macro Software Sizing and Estimating Problem. IEEE Transactions on Software Engineering. 1978; 4(4):345-61. https://doi.org/10.1109/TSE.1978.231521
5. Kemerer CF. An empirical validation of software cost estimation models. Communication of the ACM. 1987;30(5):416-29. https://doi.org/10.1145/22899.22906
6. Chandrasekaran R, Kumar R. On the Estimation of the Software Effort and Schedule using Constructive Cost Model-II and Function Point Analysis. International Journal of Computer Applications. 2012; 44(9):1-3844.
7. Khoshgoftaar M, Bullard LA, Gao K. Detecting outliers using rule-based modeling for improving CBR-based software quality classification models. Case-Based Reasoning Research and Development. 2003; p. 216-30. https://doi.org/10.1007/3-540-45006-8_19
8. El-Zaghmouri BM, Abu-Zanona MA. Fuzzy C-Mean Clustering Algorithm Modification and Adaption for Application. World of Computer Science and Information Technology Journal. 2012; 2(1):42-5.
9. Lin CT, Tsai HY. Hierarchical Clustering Analysis Based on Grey Relation grade. Information and Management Sciences. 2005; 16(1):95-105.
10. Wong CC, Chen CC. Data clustering by grey relational analysis. Journal of Grey System. 1988; 10(3):281-8.
11. Pedrycz W, Succi G, Reformat M, Musilek P, Bai X. Self organizing maps as a tool for software analysis. Canadian IEEE Conference on Electrical and Computer Engineering. 2001; 1:93-7. https://doi.org/10.1109/CCECE.2001.933665
12. Briand C, Melo WL, Wust J. Assessing the applicability of fault-proneness models across object-oriented software projects. IEEE Transactions on Software Engineering. 2002; 28(7):706-20. https://doi.org/10.1109/TSE.2002.1019484
13. Srichandan S. A new approach of Software Effort Estimation Using Radial Basis. International Journal on Advanced Computer Theory and Engineering (IJACTE). 2012; 1(1):113-20.
14. Idri A, Abran A, Mbarki S. An Experiment on the Design of Radial Basis Function Neural Networks for Software Cost Estimation. 2nd IEEE International Conference on Information and Communication Technologies. 2006; 1:230-5. https://doi.org/10.1109/ICTTA.2006.1684625
15. Idri A, Zahi A, Mendes E, Zakrani A. Software Cost Estimation Models Using Radial Basis Function Neural Networks. International Conference on Software Process and Product Measurement. 2007; p. 21-31.