ISSN:0975-9646

An Enhanced Function Point Analysis (FPA) Method for Software Size Estimation

Nilesh Choursiya[#1], Rashmi Yadav[*2]

[#]*M-Tech Student,* [*]*AssociateProfessor*
[1,2]*Department of Computer Science & Engineering*
Acropolis Technical Campus Indore

***Abstract*— To create good software we required good software size estimation tool or method because software size is a most vital component. Hence, we can enhance the accuracy of software by improving the exactness of software size estimation and it ultimately results in enhanced software effort estimates and software cost estimates. We can use these estimates for the purpose of- budgeting, staffing, scheduling, planning of software projects, etc. We are having merely high level software project requirements, for the estimation of these estimates. Really its very motivating work to produce correct software size estimates with the help of this high level knowledge. Here, we suggest a new function point based method with a new general system property/ characteristic for the accurate software size estimation. This new general system characteristic contains some new features for the intermediate and expert users. In this new characteristic we included eighteen elements that directly or indirectly affect the size of the software and ultimately to its estimation. The experimental outcomes show that the offered technique performs better than the existing function point techniques.**

***Keywords*— Software size estimation, Software effort estimation, Software cost estimation, Function point, Line of code.**

I. INTRODUCTION

In the last decade the era of software project failures gains high importance in the field of software engineering. From the foundation of computer to till date the estimation of software size and ultimately the estimation of software cost are very significant but difficult task. As the size and significance of software projects is increases, then we have to necessitate more exercise for providing exactness in software cost estimation. The calculation of cost of the all resources which will be essential for completion of the task of the software project is known as software cost estimation. For exact estimation, management, efficiency in terms of cost and effort and determination of throughput of projects, software size estimation is very critical factor [1] [2].

A number of metrics are suggested in literature by different researcher's like- line of code (LOC), feature point, use case point (UPC), object points and function points for accurate calculation of software size [3] [4].

In LOC method, the calculation of software size is done with the help of counting the number of instructions/lines in a given software program. Methods based on LOC are very simple but not very effective in terms of large size software projects [3] [4] [8]. Feature point method is drawn out from function point metrics, which employed the weight of algorithm (weight ranged from 1 to 10, 1 for simple algorithm and 10 for difficult algorithm). With the help of weighted addition of the algorithms along with function points we can calculate the feature points [5]. The computations of use case points are done from analyzing use-case system, by taking its extent with use cases in the initial stages of object oriented software project. The point count of each and every use-case is creating in the range as easy, average or hard [6]. The object point metric utilizes different ways to find out the size of the software and this extent is depends upon the quantity and intricacy of tailing objects. The object point metric is new one and easy to usage but not well-liked [5]. The function point metric is suggested by Allan Albrecht for the calculation of functionalities of the software project [7].

The computation of function point is done into three steps. The sizing of function point is depends on the five features of the software projects which are- interfaces, files, inquiries, outputs and inputs, and this features are further categorized as simple, average, or complex. Initially, on the basis of this five feature complexity we compute the amount of unadjusted function point (UFC). Then the computation of function point proceeds via an adjustment level, in which we find the value adjustment factor (VAF) which is depends upon a collection of GSC. These general systems characteristics are rate according to the weightage of each characteristic for a software project, this range is known as degree of impact and it varies from 0 to 5. At the end we can compute the function point as follows [9] [10]:

FP=UFC*VAF

In this paper, we suggest a new function point based method with a new general system property/ characteristic for the accurate software size estimation. This new general system characteristic contains some new features for the intermediate and expert users. In this new characteristic we included twenty elements that directly or indirectly affect the size of software and ultimately to its estimation.

II. LITERATURE SURVEY

Several works has been completed in the field of software size estimation in software engineering and still lots need to be complete on it. Various researchers have suggested their work in this arena from that some of the most significant works are displayed in this section.

1. *M. Pauline, P. Aruna and B. Shadaksharappa [11]:*

In this research paper authors have offered a layered model for the calculation of performance, effort and cost of given software. In this method authors discovered 3 simple but key software engineering metrics- function-points, lines of code, and person-months in the lower layer. Here, the suggested function point is comprises in the internal layer and which is attained with the help of assembling the adjustment factors. Authors employed fuzzy logic method for the computation of the quality of necessities and they added it as a novel adjustment factor. The fuzzy logic dependent method upgrades the general system characteristics and ultimately it enhances the computation of effort of the software.

2. *Bingchiang Jeng, Dowming Yeh, Deron Wang, Shu-Lan Chu and Chia-Mei Chen [12]:*

In this research paper authors have offered a function point analysis model, in which they utilized a novel technique that alters and make this model simpler, so that everyone use it in easy way. The suggested method in this paper restates the function type classifications which depend on the target application's features and on the system's design in the proposed function point analysis model. With the help of this technique we can makes the function types appropriate for the specific application area and also make it possible for the programmers they themself calculate the function point instead of taking the help from experts.

3. *Tharwon Arnuphaptrairong [13]:*

In this research paper author has suggested a Data Flow Diagram (DFD) based technique with function point analysis to deal with the application in which timing is critical problem at the initial phase of the software project development process. Author shows that, the DFD can be utilized to illustrate the functionality of the software, at the requirement phase. Here, to get the value of function points authors utilized DFD that assist as the foundation for software project effort estimation. The technique suggested by the author was confirmed with the help of the graduate student software projects at the Chulalongkorn University Business School.

4. *Dr. N. Balaji, N. Shivakumar and V. Vignaraj Ananth [14]:*

In this research paper authors have offered a combined and efficient technique which utilizes both lines of code (LOC) and Function Points (FP) for estimation of effort of software project. The authors shows that in direct technique the deviation among the real and the calculated effort is more and in indirect technique which utilized the algorithmic method COCOMO model which decreases the comparative errors and the mean absolute comparative errors. The authors also show that the estimation method which depends on function points also survives the deviation among the real and the calculated effort. Therefore, the same task of effort estimation can be employed with the help of the non-algorithmic techniques like- fuzzy logic. In this paper, authors offered triangular membership function which is based on the fuzzy logic for the estimation of effort of the software projects.

5. *Jyoti G. Borade and Vikas R. Khalkar [15]:*

In this research paper authors have demonstrated some present techniques for software effort & software cost assessment and their facets are discoursed. Software metrics which are utilized for software project size assessment & software cost assessment are also described by the authors. The author in this paper, précises past works on software cost & software size assessment. They have also discussed the pros and cons of the software size & effort assessment models and depiction of research developments in software cost & effort assessment.

EXISTING METHOD

In the existing method of function point calculation there are fourteen general system properties have presented, which are shown in [15]. In this paper we suggested a new general system property which contains some new features for the intermediate and expert users. In this new characteristic we included eighteen elements that directly or indirectly affect the size of software and ultimately to its estimation. The GSC's are utilized for the computation of adjustment factor which ultimately utilized for the computation of function point.

PROPOSED METHOD

In this paper, we suggest a new function point based method with a new general system property/ characteristic (GSC) for the accurate software size estimation. This new general system characteristic contains some new features for the intermediate and expert users. In this new characteristic we included eighteen elements that directly or indirectly affect the size of software and ultimately to its estimation. The GSC's are utilized for the computation of adjustment factor which ultimately utilized for the computation of function point. The proposed GSC is shown in table- I.

Table I

S No	Facility for Intermediate and Expert User
1	Programming by examples
2	Making throw away codes
3	User friendly codes
4	Validation features
5	Personnel safety
6	Robustness
7	Tools for analyzing
8	Error detection tools
9	Alerts for updation
10	Work on multi windows
11	Parallel Computing
12	Testable codes
13	Facility to online support like live chat, online help
14	Self-effectiveness: High sense of control over the environment
15	Flexible codes
16	powerful debugging and tracing facility
17	Scalability features
18	Easiness of Maintenance

III. RESULT

The computation of function point is done into three steps. The sizing of function point is depends on the five features of the software projects which are- interfaces, files, inquiries, outputs and inputs, and this features are further categorized as simple, average, or complex. Initially, on the basis of this five feature complexity we compute the amount of unadjusted function point (UFC). Then the computation of function point proceeds via an adjustment level, in which we find the value adjustment factor (VAF) which is, depends upon a collection of GSC, in this step we add a new general system characteristic. These general systems characteristics are rate according to the weightage of each characteristic for a software project, this range is known as degree of impact and it varies from 0 to 5. At the end, we can compute the function point as follows:

FP=UFC*VAF

The degree of impact of above 18 items of intermediate/expert user properties for given software project will be as follows:

Table II

Degree of Impact	Value
0	None
1	1<S. No<3
2	4<S. No<6
3	7<S. No<9
4	10<S. No<12
5	13<S. No<18

Now, we assume the following inputs for experimental purpose:

Inputs	Value	Weight	Value
Internal Logical Files (ILF)	2	Low	7
External Interface Files (EIF)	2	Average	7
External Inputs (EI)	3	High	6
External Outputs (EO)	3	Low	4
External Queries (EQ)	4	Average	4

TDI=42 & New TDI=45

Then, the value of function point by utilizing existing method is computed as follows:

Unadjusted Function Point (UFP) = 2*7 + 2*7 + 3*6 + 3*4 + 4*4 = 74

Value Adjustment Factor (VAF) =0.65 + TDI/100 = 0.65 + 42/100 = 1.07

Existing Function Point (FP) = UFP * VAF = 74 * 1.07 = 79.18

Where TDI is the total degree of impact

Now, we find the value of function point by utilizing suggested method is as follows:

Unadjusted Function Point (UFP) = 2*7 + 2*7 + 3*6 + 3*4 + 4*4 = 74

Value Adjustment Factor (VAF) =0.65 + New_TDI/100 = 0.65 + 45/100 = 1.10

New Function Point (FP) = UFP * VAF = 74 * 1.10 = 81.40

Where New_TDI is the total degree of impact of suggested method.So, the experimental results shows that the suggested method produces improved function point value as compared to the existing method.

IV. CONCLUSION

The accurate software size estimation of a software project is really a difficult task, if the code of the software is large enough. The exact sizing of software is very essential component for the software development phase, since it ultimately decides the cost of the given software project. With the help of general system property we can improve the resultant value of function point and ultimately enhance the quality of software size estimation. In this paper, we offer a method for the purpose of accurate software size estimation by proposing a new general system property. The experimental results, shows that the suggested method produces improved function point value as compared to the existing method

REFERENCES

[1] M. Pauline, P. Aruna and B. Shadaksharappa, "Software Cost Estimation Model based on Proposed Function Point and Trimmed Cost Drivers Using Cocomo II", International Journal of Engineering Research & Technology (IJERT), Vol. 1 Issue 5, July - 2012.

[2] http://www.slideshare.net/hemanthraj5439/introduction-to-software-cost-estimation.

[3] Rajib Mall, "Fundamentals of Software Engineering", pp. 38-53.

[4] Chander Diwaker and Astha Dhiman, "Size and Effort Estimation Techniques for Software Development", International Journal of Software and Web Sciences (IJSWS), 2013.

[5] Hareton Leung, Zhang Fan, "Software cost estimation", Department of Computing, The Hong Kong Polytechnic University, pp. 2-12.

[6] Bogdan Stępień, "Software Development Cost Estimation Methods and Research Trends", Computer Science, Vol. 5, 2003, pp. 68-82.

[7] Vahid Khatibi and Dayang N. A. Jawawi, "Software Cost Estimation Methods: A Review", Vol. 2, no. 1, ISSN 2079-8407, pp. 22-24.

[8] Nilesh Choursiya and Rashmi Yadav "A survey on Software Size and Effort Estimation Techniques " Cognitive Technical Research Journal Vol:2 Issue:2 Jul-Dec 2014

[9] Azath. H, Wahidabanu R. S. D., "Function Point: A Quality Loom for the Effort Assessment of Software Systems", International Journal of Computer Science and Network Security, IJCSNS Vol.8 No12, Dec 2008.

[10] Charles R. Symons, "Function Point Analysis: Difficulties and Improvements", IEEE Trans., Software Eng., Vol. 14, no. 1, pp.2-11, Jan 1988.

[11] M. Pauline, P. Aruna and B. Shadaksharappa, "Layered Model to Estimate Effort, Performance and Cost of the Software Projects", International Journal of Computer Applications (0975 – 8887) Volume 63– No.13, February 2013.

[12] Bingchiang Jeng, Dowming Yeh, Deron Wang, Shu-Lan Chu and Chia-Mei Chen, "A Specific Effort Estimation Method Using Function Point", Journal of information science and engineering 27, 1363-1376 (2011).

[13] Tharwon Arnuphaptrairong, "Early Stage Software Effort Estimation Using Function Point Analysis: An Empirical Validation", INTERNATIONAL JOURNAL OF DESIGN, ANALYSIS AND TOOLS FOR INTERGRATED CIRCUITS AND SYSTEMS, VOL. 4, NO. 1, DECEMBER 2013

[14] Dr. N. Balaji, N. Shivakumar and V. Vignaraj Ananth, "Software Cost Estimation using Function Point with Non Algorithmic Approach", Global Journal of Computer Science and Technology Software & Data Engineering, Volume 13, Issue 8, Version 1.0, Year 2013.

[15] Jyoti G. Borade and Vikas R. Khalkar "Software Project Effort and Cost Estimation Techniques", International Journal of Advanced Research in Computer Science and Software Engineering 3(8), August - 2013, pp. 730-739.