ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 6, June 2013

Accurate Software Size Estimation Using the Updated Function Point Analysis Model

Vikas Kumar [1], Sweta Pandey [2]

Computer Science and Application, Thapar University, Patiala, India [1]
Information Technology, Banasthali University, Jaipur, India [2]

Abstract: In this paper; a new Function Point Analysis model has been proposed. In this proposed model, a new general system characteristic is added. The expert user programming also affects the size of software. By including it in the list of general system characteristics, it creates a provision for taking end user facilities into account, while estimating the size of a project. It is clear that proposed FPA provides more accurate size estimates and it will narrow the gap between size estimated and actual size. This will result in more accurate effort and cost estimates, which ultimately results in increased productivity and proper staffing, planning and scheduling.

Keywords: FPA, cost estimation, effort, size of project

I. Introduction

This document describes the Function point analysis which measures software by quantifying the functionality the software provides to the user based primarily on logical design. Here in this Function Point Analysis model has been proposed which creates a provision for taking end user facilities into account, while estimating the size of a project. This paper comprises of four sections including the present one which describes the goal of this paper. Section II shows research based papers which illustrates related work in function point analysis. Section III gives a brief introduction regarding proposed model and experimental result. And at last section IV describe the conclusion and references.

.

II Related Work

Albrecht et al. [1] describes Function Point Analysis (FPA) method as an alternative to code-based sizing methods.

Gaffney et al. [2] illustrate international Function Point Users Group (IFPUG), a non-profit organization, which was later established to maintain and promote the practice.

IFPUG [3] [4] describes extended and also published several versions of the FPA Counting Practices Manual to standardize the application of FPA.

Symons et al. [5] describe other significant extensions to the FPA method have been introduced and widely applied in practice, such as Mark II FPA and COSMIC-FFP.

Abran et al. [6] illustrate COSMIC-FFP which is also a extension to the FPA.

N. E. Fenton et al. [7] proposed Function Point Analysis (FPA) model which consists of two main parts in the measurement. In the first part following functionalities are counted while counting the function points of the system.

- *Data Functionality*

1. Internal Logical Files (ILF)
2. External Interface Files (EIF)

- *Transaction Functionality*

1. External Inputs (EI)
2. External Outputs (EO)
3. External Queries (EQ)

Boehm et al. [8] illustrate these characteristics which contribute to Value Adjusted Factor (VAF). The final function point count is obtained by multiplying the VAF times the Unadjusted Function Point (UAF).

Symons et al. [9] describes 14 GSC's these are as follows : Data communication, Distributed functions, performance, heavily used configuration, transaction rate, online data entry, End user efficiency, Online Update, complex processing, reusability, installation ease, multiple sites, facilitate change .

.

III Proposed Enhancement in FPA

The standard equation for estimation

➢ FP = UFP * VAF
➢ Where UFP = Unadjusted Function Point and

➢ VAF = Value Adjusted Factor As mentioned, the total number of UAF is accumulated from five components.

➢ The simplified equation is as follows:

➢ UFP = EI + EO + EQ + ILF + EIF.

➢ The weights are assigned to each component based on transactional and data function types. For VAF, it is calculated from the summation of 14 GSCs as in:

♦ VAF = 0.65 + TDI/100

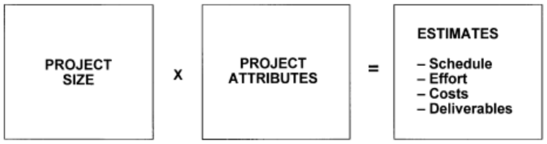

Fig 1Software-estimating principle

Software size estimation is a critical activity. If the software contains expert user facilities then the size of software increases. But the expert user facilities are not taken into account while estimating the size of software. There is always a scope to introduce new characteristics for efficient size estimation. Here we are introducing expert user programming as a new general system characteristic.

➢ VAF = 0.65 + TDI/100

• Here TDI is the sum of all 15 General System Characteristics.

• All the details of our new general system characteristic is given below.

• To evaluate expert user programming characteristic following 15 elements are used.

The degree of influence of above 15 items will be computed as follows:

Table1 1 Elements for expert user programming characteristics

S No	Expert User Facility in the Software
1	Programming by example
2	Creating throw away codes
3	Creating reusable codes
4	Easily understandable codes
5	Authentication features
6	Personnel security
7	Verification
8	Tools for analysing by debugging
9	Error detection tools
10	Testable codes
11	Availability of online help
12	Self – efficacy: High sense of control over the environment
13	Flexible codes
14	Scalability features
15	Ease of Maintenance

Table 2 Degree of influence for expert user characteristics

Degree of Influence	Description
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< 15

➢ *Experimental Results:*

• Consider the following inputs:

3. Internal Logical Files (ILF) - 02 and weight low = 7

4. External Interface Files (EIF) - 02 and weight avg = 7

5. External Inputs (EI) - 03 and weight high = 6

6. External Outputs (EO) - 03 and weight low = 4

7. External Queries (EQ) - 04 and weight avg = 4

➢ TDI = 42 & New TDI = 45

➢ Then

➢ FPA

UFP = 2*7 + 2*7 + 3*6 + 3*4 + 4*4 = 74
VAF = 0.65 + TDI/100 = 0.65 + 42/100 = 1.07
FP = UFP * VAF = 74 * 1.07 = 79.18

➢ Proposed FPA

UFP = 2*7 + 2*7 + 3*6 + 3*4 + 4*4 = 74
VAF = 0.65 + New TDI/100 = 0.65 + 45/100 = 1.10
New FP = UFP * VAF = 74 * 1.10 = 81.40

IV CONCLUSION

From above results, it is clear that proposed FPA provides more accurate size estimates. It will narrow the gap between size estimated and actual size. This will result in more accurate effort and cost estimates. This ultimately results in increased productivity and proper staffing, planning and scheduling.

REFERENCES

1. Albrecht A.J. (1979), "Measuring Application Development Productivity," Proc. IBM Applications Development Symp., SHARE-Guide, pp. 83-92.
2. Albrecht A.J. and Gaffney J. E. (1983) "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," IEEE Transactions on Software Engineering, vol. SE-9, no. 6, November
3. IFPUG (1999), "IFPUG Counting Practices Manual - Release. 4.1," International Function Point Users Group, Westerville, OH
4. IFPUG (2004), "IFPUG Counting Practices Manual - Release. 4.2," International Function Point Users Group, Princeton Junction, NJ.
5. Symons C.R. (1988) "Function Point Analysis: Difficulties and Improvements," IEEE Transactions on Software Engineering, vol. 14, no. 1, pp. 2-11
6. Abran A., St-Pierre D., Maya M., Desharnais J.M. (1998), "Full function points for embedded and real-time software", Proceedings of the UKSMA Fall Conference, London, UK, 14.
7. N. E. Fenton and S. L. Pfleeger, 1997. Software Metrics: A Rigorous and Practical Approach, 2nd Edition Revised ed. Boston: PWS Publishing.
8. Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R., and Selby, R. Cost models for future software life cycle processes: COCOMO 2.0. Annals of Software Engineering, Special Volume on Software Process and Product Measurement (1995).
9. http://www.devshed.com/c/a/Practices/An-Overview-of- Function-Point-Analysis/3