C.V.S.R.SYAVASYA / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 2, Issue 5, September- October 2012, pp.1732-1735

An Approach To Estimate Function Point Analysis Using Unadjusted Function Points And Value Adjustment Factor

C.V.S.R.SYAVASYA
Master of Technology Gitam University, Visakhapatnam, India

ABSTRACT

Function point analysis was developed by Albercht. In this approach, we calculate unadjusted function point by obtaining formulated parameter values for all the five parameters namely: ILF, EIF, EO, and EQ, EI based on the complexity and assign the value for each. After calculating the five parameters, the UAF value is obtained. On the other side, we calculate value adjustment factor by observing fourteen different characteristics and assigning a value to each characteristic based on its degree of influence. Finally unadjusted function point is multiplied by value adjustment factor which results in Function point analysis. The main reason in carrying out this approach as it reduces the risk of "inflation" of the created lines of code, and thus reducing the value of the measurement system, if developers are incentivized to be more productive. FP advocates refer to this as measuring the size of the solution instead of the size of the problem.

Keywords - External Interface Files Internal Logic Files, Unadjusted Function points, Value Adjusted Function points, Function Point Count

I. INTRODUCTION

Function point analysis is a popular method for estimating and measuring the size of application software on the functionality of the software from the user's point of view. Through this method, the size of an application system's functionality is calculated in terms of Function Point Count.

[1]Allan J Albrecht [6] of IBM proposed Function point Count as a size measure in the late 1970s. Systems continue to grow in size and complexity, becoming increasingly difficult to understand. As improvements in coding tools allow software developers to produce larger amounts of software to meet ever-expanding user requirements, a method to understand and communicate size must be used. A structured technique of problem solving, function point analysis is a method to break systems into smaller components, so they can be better understood and analyzed. This book describes function point analysis and industry trends using function points. Human beings solve problems by breaking them into smaller, understandable pieces. Problems that may initially appear to be difficult are found to be simple when dissected into their components, or classes. When the objects to be classified are the contents of software systems, a set of definitions and rules, or a scheme of classification, must be used to place these objects into their appropriate categories. Function point analysis is one such technique: **FPA is a method to break systems into smaller components, so they can be better understood and analyzed**. It also provides a structured technique for problem solving. Function Point Analysis is a structured method to perform functional decomposition of a software application.

Function points are a unit measure for software much like an hour is to measuring time, miles are to measuring distance or Celsius is to measuring temperature. Function Points are interval measures much like other measures such as kilometres, Fahrenheit; hours so on and so forth. Function Points measure software by quantifying its functionality provided to the user based primarily on the logical design. Frequently the term end user or user is used without specifying what is meant. In this case, the user is a sophisticated user. Someone that would understand the system from a functional perspective --- more than likely someone that would provide requirements or does acceptance testing. There are a variety of different methods used to count function point, but this book is based upon those rules developed by the Alan Albrecht and later revised by the International Function Point User Group (IFPUG). The IFPUG rules have much to be desired, so this book attempts to fill in gaps not defined by IFPUG.

1.1 OBJECTIVES OF FPA

According to IGPUG [2], the objectives of function point analysis are as follows:

- Measure functionality of a software system as seen from the user's perspective.
- Measure the size of software systems independent of technology used for implementation.
- Create a measurement methodology that is simple enough to minimize the overhead of the measurement process.
- Create a consistent measure among various projects and organizations.

II. METHODOLOGY

IDENTIFYING DATA FUNCTIONS AND TRANSACTION FUNCTIONS

2.1 External Inputs [3] [4]:

External Inputs (EI) - is an elementary process in which data crosses the boundary from outside to inside. This data is coming external to the application. The data may come from a data input screen or another application. The data may be used to maintain one or more internal logical files. The data can be either control information or business information. If the data is control information it does not have to maintain an internal logical file. If an external input adds changes and deletes information on an internal logical file, then this represents three external inputs. External inputs may be preceded by an external inquiry. Hence a full function screen is add, change, delete and inquiry

2.2 External Outputs

External Outputs (EO) - an elementary process in which **derived data** passes across the boundary from inside to outside. Additionally, an EO may update an ILF. The data creates reports or output files sent to other applications. These reports and files are created from information contained in one or more internal logical files and external interface files.

Derived Data is data that is processed beyond direct retrieval and editing of information from internal logical files or external interface files. Derived data is usually the result of algorithms, or calculations. Derived data occurs when one or more data elements are combined with a formula to generate or derive an additional data element(s). This derived data does not appear in any FTR.

2.3 External Inquiries

External Inquiry (EQ) - an elementary process with both input and output components that result in data retrieval from one or more internal logical files and external interface files. The input process does not update or maintain any FTR's (Internal Logical Files or External Interface Files) and the output side does not contain derived data. Transactions between applications should be referred to as interfaces. You can only have an external output or external inquiry of data external to your application. If you get data from another application and add it to a file in your application, this is a combination get and add (external inquiry and external input).

2.4 Internal Logic Files :

Internal Logical Files (ILF) - a user identifiable group of logically related data that resides entirely within the application boundary and is maintained through External Inputs. An internal logical file has the inherent meaning it is internally maintained, it has some logical structure and it is stored in a file. Even though it is not a rule, an ILF should have at least one external output and/or external inquiry. That is, at least one external output and/or external inquiry should include the ILF as an FTR. Simply put, information is stored in an ILF, so it can be used later. The EO or EQ could be from another application. It is worth noting that it is possible that a specific ILF is not referenced by EO or EQ, but it is used by an EI (other than the EI that maintains it). Again, even though it is not a rule, an ILF should have at least one external input.

2.5 External Interface Files

External Interface Files (EIF) - a user identifiable group of logically related data that is used for reference purposes only. The data resides entirely outside the application boundary and is maintained by other applications external inputs. The external interface file is an internal logical file for another application. An application may count a file as either an EIF or ILF not both. An external interface file has the inherent meaning it is externally maintained (probably by some other application), an interface has to be developed to get the data and it is stored in a file. Each EIF included in a function point count must have at least one external output or external interface file against it. At least one transaction, external input, external output or external inquiry should include the EIF as a FTR. Every application, which references the EIF, needs to include it in their FP Count. Some organizations have a pull theory and others have a push theory of data. The pull theory is an external application "reaching into" other applications and retrieving data. Those organizations which have push theory require applications to create interfaces (EO or EQ) which other applications read.

These 5 function types are then ranked according to their complexity: Low, Average or High, using a set of prescriptive standards. Organizations that use FP methods develop criteria for determining whether a particular entry is Low, Average or High. Nonetheless, the determination of complexity is somewhat subjective. After classifying each of the five function types, the UFP is computed using predefined weights for each function type.

III. COMPUTE UNADJUSTEDFUNCTION POINT(UFP)

The Unadjusted Function Points [5] for each function depends on the function type complexity of the function determined in the previous section. The Unadjusted function points [6] to be assigned are given in the table below:

Table-1 showing values for each parameter in UFP

Complexity	Function Type				
	ILF	EIF	EI	EO	EQ
Simple	7	5	3	4	3
Average	10	7	4	5	4
Complex	15	10	6	7	6

The total **UFP** is determined by summation of all parameters and finally the result is carried to compute FP.

IV. VALUE ADJUSTMENT FACOR

The value adjustment factor (VAF) [6] is based on 14 general system characteristics (GSC's) [7] [8] that rate the general functionality of the application being counted. Each characteristic has associated descriptions to determine the degrees of influence. The degrees of influence range on a scale of zero to five, from no influence to strong influence. Each characteristic is assigned the rating based upon detail descriptions provided by the IFPUG [9] 4.1 Manual. They ratings are:

0 Not present or no influence
1 Incidental influence
2 Moderate influences
3 Average influences
4 Significant influences
5 Strong influences throughout

The degrees of influence rating of each General System Characteristics are added to give Total Degree of Influence (TDI). Therefore VAF [10] is computed as,

VAF = (TDI * 0.01) + 0.65

The value of TDI ranges from a minimum of 0 to a maximum of 70. As a result the value of VAF can range from 0.65 to 1.35, where the mid-point is 1.

V. Computing the Function Point Count:

For the development project, the function point count is computed as,

FPC = UFP * VAF

The above is the formula used to compute the Function point count.

VI. RESULTS AND DISCUSSION

PROJECT-1:

In this project taken as example, we take values of Estimated Count for each parameter (EI, EO, EQ, ILF, and EIF) and multiply those values to the weight of function type according to table-1.

Table-2 showing the results of parameters in Function point analysis of to calculate UFP

EI	12*(Average=4)	48
EO	7*(Average=5)	35
EQ	10*(Average=4)	40
ILF	5*(Average=10)	50
EIF	3*(Average=7)	21
	TOTAL UFP=	194

In the above table, it is observed that, for Estimated count 12 taken for EI is calculated based on Average Function type for EI which is given as 4 according to table-1. Hence the FP-Count for EI is 48. In case of EO, for Estimated count 7 taken for EO is calculated based on Average Function type for EO which is given as 5 according to table-1. Hence the FP-Count for EO is 35. In the same way the remaining FP-Count values are calculated and finally the total Unadjusted Function Point (**UFP**) is obtained as **194**.

To calculate TDI, the 14 General characteristics are values according to their degree of influence from 0 to 5. For 0 there is no influence, 1= Incidental, 2 = Moderate, 3 = Average, 4 = Significant, 5= Essential

The General Characteristics for 0-10 are assigned as 0 and from 11 to 14 are assigned as 4. Finally, The Total Degree of Influence (TDI) is resulted as 16.

Therefore,

V.A.F = (T.D.I * 0.01) + 0.65
= (16 *0.01) +0.65
= 0.81

Value adjustment factor which is calculated in the above is obtained as 0.81 is one of the parameter obtained to obtain the function point count [7]. The result obtained by value adjustment factor is multiplied by the unadjusted function point to result in function point count (FPC) which is shown as follows:

Finally, FPC = UFP * VAF
= 194*0.81
= 157

The Function Point Count [12] is obtained as 157 is the final result of this project which evolved after detailed calculation of the obtained necessary parameter values.

VII. CONCLUSION

The main objective of taking up this project is to explore the new results in the field of Size Estimation as software size is one of the important software attribute. The fast growth of software development is allowing maintenance of large

repositories of project related information which stores information about software size. Value adjustment factor here in this case plays a major role in evolving the final output after computing randomly assigned values to find out total degree of influence. This final output tends to be a part of the basic formulae for computing the function point count. Function points can be used to size software project [7] applications accurately, as sizing is an important factor in determining productivity.

Since function point has a unique and consistent method, different people measuring them will give almost the same result with very little margin of error. A non-technical person can easily understand function points, which helps in communicating the same to the end-user effectively and easily. This work can be extended further by taking up different rating projects to general characteristics according to its system influence and by applying new functionalities of estimation count under unadjusted function points, which tends to innovating new results of function point analysis into action.

REFERENCES

1. Software Requirements and Estimation by Estimation by Swapna Kishore and Rajesh Naik published in 2001 by Tata McGRAW HILL Company Limited.
2. "Function Points: A Study of Their Measurement Processes and Scale Transformations", J. System Software, 1994; 25:171-184
3. http://www.ifpug.org/?page_id=10
4. http://www.compaid.com/caiinternet/ezine/garmus-functionpointintro.pdf
5. Abran, A., and Robillard, P. N., Identification of the structural weakness of function point metrics, in 3rdAnnual Oregon Workshop on Software Metrics, March 18, 1991.
6. Albrecht, A. J., and Gaffney, J. E., Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation, IEEE Trans. Software Eng. SE-9, 639-648 (1983).
7. Behrens, C. A., Measuring the Productivity of Computer Systems Development Activities with Function Points, IEEE Trans. Software Eng., SE-9, 648-652 (1989).
8. IFPUG, http://www.ifpug.org/
9. Dunn, R. H., Software Quality-Concepts and Plans, Prentice-Hall, Englewood Cliffs, New Jersey, 1990.
10. Eijogu, L. O., Beyond Structured Programming: An Introduction to the Principles of Applied Software Metrics, J. Struct. Progr. 11 (1990).
11. Eijogu, L. O., Software Engineering with Formal Metrics, QED Information Sicences, Wellesley, Massachusetts, 1991.
12. Low, G. C., and Ross, J. D., Function Points in the Estimation and Evaluation of the Software Process, IEEE Trans. Software Eng. 16, 64-71 (1990).