International Journal of Scientific Engineering and Applied Science (IJSEAS) – Volume-2, Issue-1, January 2016
ISSN: 2395-3470
www.ijseas.com

Estimation of Lines of Code using Funtion Point Analysis – Prior to the Implementation

Dr. B V Ramana Murthy,
Department of Computer Science and Engineering, MIST,Hyderabad, India
drbvrm@gmail.com

Software measurement [1], once an obscure and esoteric specialty, has become essential to good software engineering [2, 3, 4]. Many of the best software developers measure characteristics of the software to get some sense of whether the requirements are consistent and complete, whether the design is of high quality, and whether the code is ready to be tested. Effective project manager's measure attributes of process and product to be able to tell when the software will be ready for delivery and whether the budget will be exceeded. Informed customers measure aspects of the final product to determine if it meets the requirements and is of sufficient quality. And maintainers must be able to assess the current product to see what should be upgraded and improved.Here, the present study of metrics is based for function oriented software.
Keywords : FPA, metrics, size estimations, code estimation.

Metrics for the Analysis Model:- These metrics address various aspects of the analysis model and include:

- Functionality Delivered: Provides an indirect measure of the functionality that is packaged with in the software.
- System Size: measures[5] of the overall size of the system defined in terms of information available as part of the analysis model[7].
- Specification Quality: provides an indication of the specificity and completeness of a requirements specification.

Function-Based Metrics: The Function Point Metric [FP] [7, 8, 9, 10], first proposed by Albrecht, can be used effectively as a means for measuring[6] the functionality delivered by a system. Using historical data, the FP can then be used to

(1) estimate the cost or effort required to design, code, and test the software;

(2) predict the number of errors that will be encountered during testing.

(3) forecast the number of components and/or the number of projected source lines in the implemented system.

Function points are delivered using an empirical relationship based on countable (direct) measures of software's information domain and assessments of software complexity. Information domain values are defined in the following manner:

Number of external inputs (EIs): Each external input originates from a user or is transmitted from another application and provides distinct application-oriented data or control information. Inputs are often used to update internal logical files (ILFs). Inputs should be distinguished from inquiries, which are counted separately.

Number of external outputs (EOs): Each external output is derived within the application and provides information to the user. In this context external output refers to reports, screens, error messages, and so on. Individual data items within a report are not counted separately.

Number of external inquiries (EQs): An external inquiry is defined as an online input that results in the generation of some immediate software response in the form of an on-line output (often retrieved from an ILF).

Number of internal logical files (ILFs): Each internal logical file is a logical grouping of data that resides within the application's boundary and is maintained via external inputs.

Number of external interface files (ELFs): Each external interface file is a logical grouping of data that resides external to the application but provides data that may be of use to the application.

Once these data have been collected, the table in figure 2.1 is completed and a complexity value is associated with each count. Organizations that use function point methods develop criteria for determining whether a particular entity is simple, average, or complex. Nonetheless, the determination of complexity is somewhat subjective.

To compute function points (FP), the following relationship is used:
FP = count total * [0.65 + ∑ (Fi)]

Where count total is the sum of all FP entries obtained from below figure
The Fi (i = 1 to 14) are value adjustment factors (VAF) based on responses to the following questions :

Computing Function Points	Information Domain Value	Count	Weighting Factor Simple	Average	Complex	
	External Inputs (EIs)	value *	3	4	6	=
	External Outputs (EOs)	value *	4	5	7	=
	External Inquiries (EQs)	value *	3	4	6	=
	Internal Logical Files (ILFs)	value *	7	10	15	=
	External Interface Files (EIFs)	value *	5	7	10	=
	Count Total					=
FP Computing						

1. Does the system require reliable backup and recovery?
2. Are specialized data communications required to transfer information to or from the application?
3. Are there distributed processing functions?
4. Is performance critical?
5. Will the system run in an existing, heavily utilized operational environment?
6. Does the system require on-line data entry?
7. Does the on-line data entry require the input transaction to be built over multiple screens or operations?
8. Are the ILFs updated on-line?
9. Are the inputs, outputs, files, or inquires complex?
10. Is the internal processing complex?
11. Is the code designed to be reusable?

12. Are conversion and installation included in the design?
13. Is the system designed for multiple installations in different organizations?
14. Is the application designed to facilitate change and for ease of use by the user?

Research Proposal :- To estimate LOC – Lines of Code using FPA- Function Point Analysis

Step1 Collect LOC of function oriented projects ex: C language , different project types and sizes
Step2 Calculate FPA of each individual project using code
Step 3 Extend to the data set, maintain tabular column FP vs LOC
Step 4 Apply Linear Regression - LOC=a+b*FP
Where a and b are constants , find a and b from step 2
Step 5 Using step 4 , once the FP of the project is known then LOC can be estimated prior to implementation

Theoretical validity: The theoretical validation process attempts to verify that whether function point metric measures the internal attribute that is supposed to, namely, the size of the software product being considered. The system design under counting can be seen as a set of domain values between them.

Predictive validity: Once the theoretical consistency of the function point metric has been verified, the usefulness of the method has to be proven by means of an empirical validation process. The aim of the empirical validation is, to determine if the metric is related to a specific quality factor (e.g., maintainability, reliability, efficiency). The empirical validation procedure depends on the quality functionalities, that are supposed to use the metric. The quality functionalities are quality tasks, performed to achieve a target quality level. Let FAT2 and FPT2 be the actual and predicted values of a time T2 respectively. Metric M can predict the quality factor F with the required accuracy if $|$ (FAT2 - FPT2)/ FAT2 $|$ < A, where A represents the prediction error threshold, for a representative sample set of software systems. A descriptive analysis both for the dependent variable (LOC) and the independent variable (FP). The linear regression between FP and LOC leads to LOC = a + b * FP where a and b are constants. The linear regression allows us to determine the equation of line, which interpolates data and can be used to predict the lines of code during the design phase.
By regression we mean average relationship between two variables and this relationship is used to estimate or predict the most likely values of one variable for specified value of the other variable. The latter is called independent or the explanatory variable and the other is called dependent or the explained variable. “Regression is the measure of the average relationship between two or more variables in terms of the original units of the data”.

Regression Lines:-The regression line is a graphical device to describe the average relationship between the two variables. It indicates the law of changes in the mean value of one variable corresponding to the mean value of the other. If the bi-variate data are plotted as points on a graph paper, it will be found that the concentration of points follows a certain pattern, showing some relationship between the variables. The lines of regression are straight lines drawn through these points which indicate the average values of one variable for given or known values of the other variable. For

two variables X and Y, there are two lines of regression:

(1) Regression line of X on Y, and
(2) Regression line of Y on X.

The former is used to find the best estimates of X (dependent variable) for given values of Y (independent variable) and the latter, to find the best estimates of Y (dependent variable) for given values of X (independent variable). The two lines of regression always intersect at the point (x, y). They are distinct, unless the two variables are perfectly correlated. If X and Y are uncorrelated, the two lines of regression are at right angles to each other.

Regression Equations:-The regression lines are algebraically expressed by regression equations. The regression equations may be regarded as expressions for estimating from a given value of the independent variable the average corresponding value of the independent variable. Since there are two regression lines, there are two regression equations: one to measure the regression of Y on X and another to measure the regression of X on Y. Regression equations are expressed in terms of mean values of the two series and indicate the variation from the mean of other series.Regression line of X on

Y: X = a + bY where a and b are two constants (‘a’ representing the X intercept and ‘b’ represents the slope of the said line indicating the amount of change in the value of the dependent variable for a unit change in independent variable), X is the dependent variable and Y is the independent variable. Through this line we can estimate the probable value of X for any given value of Y. The two constants can be worked out for the given data through the use of normal equations, viz.,

$\Sigma X = Na + b\Sigma Y$

and $\Sigma xy = a\Sigma y + b\Sigma y2$

This line can also be expressed as:

$$(X-x) = r\, \sigma x / \sigma y\, (Y-y)$$

Regression line of Y on X: Y = c + dX, where c and d are the constants as stated above, Y is the dependent variable and X is independent variable.

VALIDATIONS

The quality of regression analysis is determined by

1. The square of the linear correlation coefficient, R2 which provides a measure of the quality of the prediction.
2. By using AI techniques applying K-fold cross validation given by Mc.Carthy and Lehnert-Chose

Step-1: Shuffle the items in the training set.
Step-2: Divide the training set into ‘k’ equal parts of size ‘n’ say, i sets of size n = i sentences.
Step-3: Do i = 1 to k times

a. call the ith set of ‘n’ sentences the test set and put it aside.
b. Train the system on the remaining (k-1) sets; test the system on the test set, record the performance.
c. Clear memory, that is, forget every thing learned during training.
d. Calculate the average performance from the k test cases.

Validations are used to show the consistent of the research under allowed threshold value.

Conclusion Estimation of size oriented metrics is useful to estimate LOC prior to the implementation of project, help for

project manager to plan the activities of the project Life cycle.

[1] Fenton, N., "Software Measurement: A Necessary Scientific Basis", IEEE Trans. Software Engineering, Vol. SE-20, no. 3, March 1994, pp. 199-206.

[2] Humphrey, W., A Discipline for Software Engineering, Addison-Wesley, 1995.

[3] Somerviue, I., Software Engineering, 6th ed. , Addison-Wesley, 2001.

[4] Gilb, T., Principles of Software Engineering Management, Addison-Wesley, 1988.

[5] Putnam, L., and W. Myers, Measures of Excellence, Yourdors Press, 1992.

[6] Felican, L., and G. Zalateu, "Validating Halstead's Theory for Pascal Programs", IEEE Trans. Software Engineering, Vol. SE-15, no. 2, December 1989, pp. 1630-1632.

[7] Albrecht, A. J., and J. E. Gaffney, "Software Function, Source Lines of Code and Development Effort Prediction: A Software Science Validation", IEEE Trans. Software engineering, November 1983, pp. 639-648.

[8] A. J. Albrecht, "Measuring application development productivity", Proc. of the Joint SHARE / GUIDE / IBM Application Development Symposium, Oct. 1979, pp. 83-92.

[9] A. J. Albrecht, J. E. Gaffney, "Software Function, source lines of code, and development effort prediction: A software science validation ",IEEE Trans. Software Eng. 9, no. 6 (1983) 639-648.

[10] J. B. Dreger, Function Point Analysis, Prentice-Hall. 1989.

Dr. B. V.Ramana Murthy has done his PhD from Osmania University, presently he working as Professor in Computer Science and Engineering, has 18 years of experience in teaching and R&D. His primary area of interest is Software Engineering & Web Engineering.