International Journal of Scientific & Engineering Research, Volume 8, Issue 4, April-2017
ISSN 2229-5518

Estimation Of Factors Affecting The Object Oriented Projects Using Vaf And Faf

S.Rajalakshmi

Abstract- The idea behind this paper includes the scope to fit for huge sized object oriented projects of automatic installation and updation of adjusted function point count.(AFPC).This concept inspire the definition of solving the complex processing of projects. It can be explained by the high phenomenon concentration of VAF (Value adjusted factor) and FAF (Final adjusted factor) i.e., the project being developed using FPC(Function point count) and enhancing it. According to Maslow's hierarchy of needs, it suggest the self-realization, self esteem, social, safety and physiological applications. The unadjusted function point count (UFPC) is rectified with adjusted function point count (AFPC).

The system includes the automatic updating of different locations – object oriented projects performance reports. The project product file and the location file is maintained by EDP(Electronic data processing).At each instance the sales file with data received is given by schematic depiction of components using FPC. We can understand the system with respect to the design structure, size and concern complexity. It is easy to substantiate the concept with more confidence of flicking the content accordingly.

Note: Updating UFPC into AFPC using VAF and FAF

FPC=UFPC * VAF (Initial state)

FPC=AFPC * VAF (Final state)

Keywords: UFPC, AFPC, VAF, FAF, FPC, EDP

1.INTRODUCTION

Quality and relevance are essential components of our projects. The pace of a specific project is based strictly on the experience that course participants bring with them and tailored to the requirements they face. A combination of group work and self-study provides the high level of flexibility needed to maximize the learning experience.

Function Point Analysis is a structured technique of problem solving. It is a method to break systems into smaller components, so they can be better understood and analyzed 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 an ordinal measure much like other measures such as kilometers, Fahrenheit, hours, so on and so forth.

S.Rajalakshmi.,MCA.,M.Phil.,
Assistant Professor,Department of BCA,
Arignar Anna College(Arts and Science)-krishnagiri

. Function Point Analysis is a structured technique of classifying components of a system. It is a method to break systems into smaller components, so they can be better understood and analyzed. It provides a structured technique for problem solving. In the world of Function Point Analysis, systems are divided into five large classes and general system characteristics.

The first three classes or components are External Inputs, External Outputs and External Inquires each of these components transact against files therefore they are called transactions. The next two Internal Logical Files and External Interface Files are where data is stored that is combined to form logical information. The general system characteristics assess the general functionality of the system.Function Point Analysis was developed first by Allan J. Albrecht in the mid 1970s. It was an attempt to overcome difficulties associated with lines of code as a measure of software size, and to assist in developing a mechanism to predict effort associated with software development.

2. OBJECTIVES OF FUNCTION POINT ANALYSIS

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. Since Function Points measures systems from a functional perspective they are independent of technology. Regardless of language, development method, or hardware platform used, the number of function points for a system will remain constant. The only variable is the amount of effort needed to deliver a given set of function points; therefore, Function Point Analysis can be used to determine whether a tool, an environment, a language is more productive compared with others within an organization or among organizations. This is a critical point and one of the greatest values of Function Point Analysis.

Function Point Analysis can provide a mechanism to track and monitor scope creep. Function Point Counts at the end of requirements, analysis, design, code, testing and implementation can be compared. The function point count at the end of requirements and/or designs can be compared to function points actually delivered. If the project has grown, there has been scope creep. The amount of growth is an indication of how well requirements were gathered by and/or communicated to the project team. If the amount of growth of projects declines over time it is a natural assumption that communication with the user has improved.

3. CHARACTERISTIC OF QUALITY FUNCTION POINT ANALYSIS

Function Point Analysis should be performed by trained and experienced personnel. If Function Point Analysis is conducted by untrained personnel, it is reasonable to assume the analysis will done incorrectly. The personnel counting function points should utilize the most current version of the Function Point Counting Practices Manual.

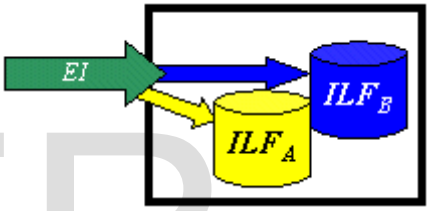
Current application documentation should be utilized to complete a ***function point count.*** For example, screen formats, report layouts, listing of interfaces with other systems and between systems, logical and/or preliminary physical data models will all assist in Function Points Analysis.

The task of counting function points should be included as part of the overall project plan. That is, counting function points should be scheduled and planned. The first function point count should be developed to provide sizing used for estimating.

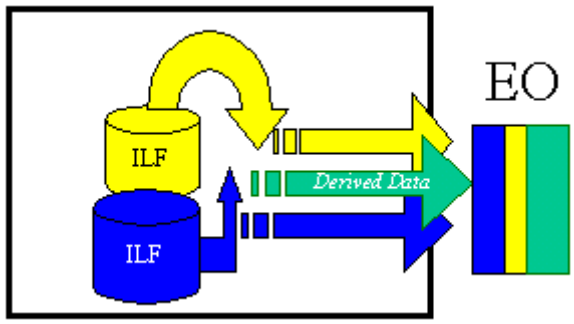
The Five Major Components

Since it is common for computer systems to interact with other computer systems, a boundary must be drawn around each system to be measured prior to classifying components. This boundary must be drawn according to the user's point of view. In short, the boundary indicates the border between the project or application being measured and the external applications or user domain. Once the border has been established, components can be classified, ranked and tallied.

External Inputs (EI) - is an elementary process in which data crosses the boundary from outside to inside. This 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 update an internal logical file. The graphic represents a simple EI that updates 2 ILF's (FTR's).

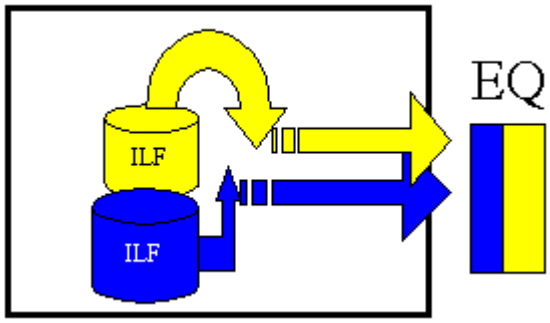


External Outputs (EO) - an elementary process in which derived data passes across the boundary from inside to outside. Additionally, an EO may update an ILF. The data creates reports or output files sent to other applications. These reports and files are created from one or more internal logical files and external interface file. The following graphic represents on EO with 2 FTR's there is derived information (green) that has been derived from the ILF's



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 any

Internal Logical Files, and the output side does not contain derived data. The graphic below represents an EQ with two ILF's and no derived data.



Internal Logical Files (ILF's) - a user identifiable group of logically related data that resides entirely within the applications boundary and is maintained through external inputs.

External Interface Files (EIF's) - a user identifiable group of logically related data that is used for reference purposes only. The data resides entirely outside the application and is maintained by another application. The external interface file is an internal logical file for another application.

After the components have been classified as one of the five major components (EI's, EO's, EQ's, ILF's or EIF's), a ranking of low, average or high is assigned. For transactions (EI's, EO's, EQ's) the ranking is based upon the number of files updated or referenced (FTR's) and the number of data element types (DET's). For both ILF's and EIF's files the ranking is based upon record element types (RET's) and data element types (DET's). A record element type is a user recognizable subgroup of data elements within an ILF or EIF. A data element type is a unique user recognizable, non recursive, field.

Each of the following tables assists in the ranking process (the numerical rating is in parentheses). For example, an EI that references or updates 2 File Types Referenced (FTR's) and has 7 data elements would be assigned a ranking of average and associated rating of 4. Where FTR's are the combined number of Internal Logical Files (ILF's) referenced or updated and External Interface Files referenced.

EI Table

FTR's	DATA ELEMENTS		
	1-4	5-15	> 15
0-1	Low	Low	Ave
2	Low	Ave	High
3 or more	Ave	High	High

Shared EO and EQ Table

FTR's	DATA ELEMENTS		
	1-5	6-19	> 19
0-1	Low	Low	Ave
2-3	Low	Ave	High
> 3	Ave	High	High

Values for transactions

Rating	VALUES		
	EO	EQ	EI
Low	4	3	3
Average	5	4	4
High	7	6	6

For both ILF's and EIF's the number of record element types and the number of data elements types are used to determine a ranking of low, average or high. A Record Element Type is a user recognizable subgroup of data elements within an ILF or EIF. A Data Element Type (DET) is a unique user recognizable, non recursive field on an ILF or EIF.

RET's	DATA ELEMENTS		
	1-19	20 - 50	> 50
1	Low	Low	Ave
2-5	Low	Ave	High
> 5	Ave	High	High

Rating	Values	
	ILF	EIF
Low	7	5
Average	10	7
High	15	10

The counts for each level of complexity for each type of component can be entered into a table such as the following one. Each count is multiplied by the numerical rating shown to determine the rated value. The rated values on each row are summed across the table, giving a total value for each type of component. These totals are then summed across the table, giving a total value for each type of component. These totals are then summoned down to arrive at the Total Number of Unadjusted Function Points.

The value adjustment factor (VAF) is based on 14 general system characteristics (GSC's) that rate the general functionality of the application being counted. Each characteristic has associated descriptions that help determine the degrees of influence of the characteristics. The degrees of influence range on a scale of zero to five, from no influence to strong

3.1 GSC -14 CHARACTERISTICS

1. Data communications
2. Distributed data processing
3. Performance
4. Heavily used configuration
5. Transaction rate
6. On-Line data entry
7. End-user efficiency
8. On-Line update
9. Complex processing
10. Reusability
11. Installation ease
12. Operational ease
13. Multiple sites
14. Facilitate change

Once all the 14 GSC's have been answered, they should be tabulated using the IFPUG Value Adjustment Equation (VAF)*where:* Ci = degree of influence for each General System Characteristic

$$VAF = 0.65 + [(Ci) / 100] *i$$

i is from 1 to 14 representing each GSC.

The final Function Point Count is obtained by multiplying the VAF times the Unadjusted Function Point (UAF).

$$FP = UAF * VAF$$

4.SUMMARY OF BENEFITS OF FUNCTION POINT ANALYSIS

Function Points can be used to size software applications accurately. Sizing is an important component in determining productivity (outputs/inputs).They can be counted by different people, at different times, to obtain the same measure within a reasonable margin of error.

Function Points are easily understood by the non technical user. This helps communicate sizing information to a user or customer.

Function Points can be used to determine whether a tool, a language, an environment, is more productive when compared with others.

5.CONCLUSIONS

Accurately predicting the size of software has plagued the software industry for over 45 years. Function Points are becoming widely accepted as the standard metric for measuring software size. Now that Function Points have made adequate sizing possible, it can now be anticipated that the overall rate of progress in software productivity and software quality will improve. Understanding software size is the key to understanding both productivity and quality. Without a reliable sizing metric relative changes in productivity (Function Points per Work Month) or relative changes in quality (Defects per Function Point) can not be calculated. If relative changes in productivity and quality can be calculated and plotted over time, then focus can be put upon an organizations strengths and weaknesses. Most important, any attempt to correct weaknesses can be measured for effectiveness.

REFERENCES:

Garmus, David and Herron, David: *"Function Point Analysis: Measurement Practices for Successful Software Projects,"* Addison-Wesley,December,2000.

Kan, Stephen H.: *"Metrics and Models in Software Quality Engineering,"* Addison-Wesley Pub Co, February, 1995.

ARTICLES:

Aguiar, Mauricio, "*Function points or Use Case Points?*" MetricViews, Summer 2009, pp 14 – 15.

Dekkers, C., *"Managing (the Size of) Your Projects – a Project Management Look at Function Points,"*CrossTalk, February 1999.

Dekkers, C., *"Demystifying Function Points: Let's Understand Some Terminology,"* IT Metrics Strategies, October 1998.

Garmus, David: *"Function Point Counting in a Real-Time Environment,"* CrossTalk, January 1996.

Heller, Roger: *"An Introduction to Function Point Analysis,"* CrossTalk, The Journal of Defense Software Engineering, Volume 8, No. 11, November/December 1995, Pages 24-26.