Introduction to Function Point Analysis

Software systems, unless they are thoroughly understood, can be like an ice berg. They are becoming more and more difficult to understand. Improvement of coding tools allows software developers to produce large amounts of software to meet an ever expanding need from users. As systems grow a method to understand and communicate size needs to be used. 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

Human beings solve problems by breaking them into smaller understandable pieces. Problems that may appear to be difficult are simple once they are broken into smaller parts -- dissected into classes. Classifying things, placing them in this or that category, is a familiar process. Everyone does it at one time or another -- shopkeepers when they take stock of what is on their shelves, librarians when they catalog books, secretaries when they file letters or documents. When objects to be classified are the contents of systems, a set of definitions and rules must be used to place these objects into the appropriate category, a scheme of classification. 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.

Brief History

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. The method was first published in 1979, then later in 1983 . In 1984 Albrecht refined the method and since 1986, when the International Function Point User Group (IFPUG) was set up, several versions of the Function Point Counting Practices Manual have been published by IFPUG. The current version of the IFPUG Manual is 4.1.

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.

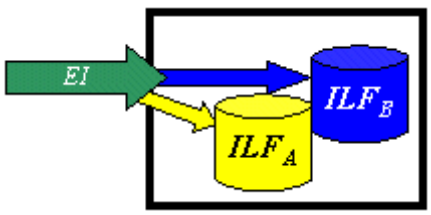
Characteristic of Quality Function Point Analysis

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.

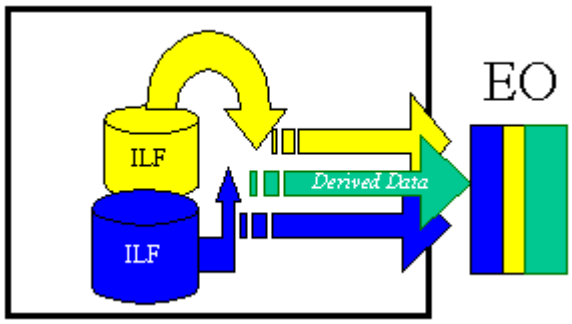
The Five Major Components

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.

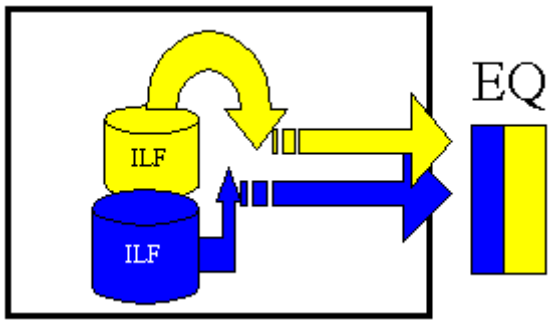
External Inputs (EI) – is an elementary process in which data crosses the boundary from outside to inside. This data may come from a data input screen or another application. The data may be used to maintain one or more internal logical files. The data can be either control information or business information. 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.

All components are rated as Low, Average or High

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

Like all components, EQâ€™s are rated and scored. Basically, an EQ is rated (Low, Average or High) like an EO, but assigned a value like and EI. The rating is based upon the total number of unique (combined unique input and out sides) data elements (DETâ€™s) and the file types referenced (FTRâ€™s) (combined unique input and output sides). If the same FTR is used on both the input and output side, then it is counted only one time. If the same DET is used on both the input and output side, then it is only counted one time.

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.

Type of Component	Complexity of Components			
	Low	Average	High	Total
External Inputs	___ x 3 = ___	___ x 4 = ___	___ x 6 = ___	
External Outputs	___ x 4 = ___	___ x 5 = ___	___ x 7 = ___	
External Inquiries	___ x 3 = ___	___ x 4 = ___	___ x 6 = ___	
Internal Logical Files	___ x 7 = ___	___ x 10 = ___	___ x 15 = ___	
External Interface Files	___ x 5 = ___	___ x 7 = ___	___ x 10 = ___	
		Total Number of Unadjusted Function Points		_____
		Multiplied Value Adjustment Factor		_____
		Total Adjusted 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 influence. The IFPUG Counting Practices Manual provides detailed evaluation criteria for each of the GSC'S, the table below is intended to provide an overview of each GSC.

General System Characteristic		Brief Description
1.	Data communications	How many communication facilities are there to aid in the transfer or exchange of information with the application or system?
2.	Distributed data processing	How are distributed data and processing functions handled?
3.	Performance	Was response time or throughput required by the user?
4.	Heavily used configuration	How heavily used is the current hardware platform where the application will be executed?
5.	Transaction rate	How frequently are transactions executed daily, weekly, monthly, etc.?
6.	On-Line data entry	What percentage of the information is entered On-Line?
7.	End-user efficiency	Was the application designed for end-user efficiency?
8.	On-Line update	How many ILF’s are updated by On-Line transaction?
9.	Complex processing	Does the application have extensive logical or mathematical processing?
10.	Reusability	Was the application developed to meet one or

		many user's needs?
11.	Installation ease	How difficult is conversion and installation?
12.	Operational ease	How effective and/or automated are start-up, back-up, and recovery procedures?
13.	Multiple sites	Was the application specifically designed, developed, and supported to be installed at multiple sites for multiple organizations?
14.	Facilitate change	Was the application specifically designed, developed, and supported to facilitate change?

Once all the 14 GSC's have been answered, they should be tabulated using the IFPUG Value Adjustment Equation (VAF) –

$$VAF = 0.65 + \left[\left(\sum_{i=1}^{14} Ci\right) / 100\right]$$

where: Ci = degree of influence for each General System Characteristic
.i = is from 1 to **14** representing each GSC.
$\sum$ = is summation of all 14 GSC's.

Another way to understand the formula is VAF = (65 + TDI)/100, where TDI is the sum of the results from each question.

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

Summary of benefits of Function Point Analysis

1. Function Points can be used to size software applications accurately. Sizing is an important component in determining productivity (outputs/inputs).
2. They can be counted by different people, at different times, to obtain the same measure within a reasonable margin of error.
3. Function Points are easily understood by the non technical user. This helps communicate sizing information to a user or customer.
4. Function Points can be used to determine whether a tool, a language, an environment, is more productive when compared with others.

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.