Disadvantages of Halstead Metrics

- It depends on the complete code.
- It has no use as a predictive estimating model.
- Limited scope: The metrics focus only on the complexity and effort required to develop and maintain a software program, and do not take into account other important factors such as reliability, maintainability, and usability.
- Limited applicability: The metrics may not be applicable to all types of software programs, such as those with a high degree of interactivity or real-time requirements.
- Limited accuracy: The metrics are based on a number of assumptions and simplifications, which may limit their accuracy in certain situations.

Functional Point (FP) Analysis

Allan J. Albrecht initially developed function Point Analysis in 1979 at IBM and it has been further modified by the International Function Point Users Group (IFPUG). FPA is used to make estimate of the software project, including its testing in terms of functionality or function size of the software product. However, functional point analysis may be used for the test estimation of the product. The functional size of the product is measured in terms of the function point, which is a standard of measurement to measure the software application.

Objectives of FPA

The basic and primary purpose of the functional point analysis is to measure and provide the software application functional size to the client, customer, and the stakeholder on their request. Further, it is used to measure the software project development along with its maintenance, consistently throughout the project irrespective of the tools and the technologies.

Following are the points regarding FPs

1. FPs of an application is found out by counting the number and types of functions used in the applications. Various functions used in an application can be put under five types, as shown in Table:

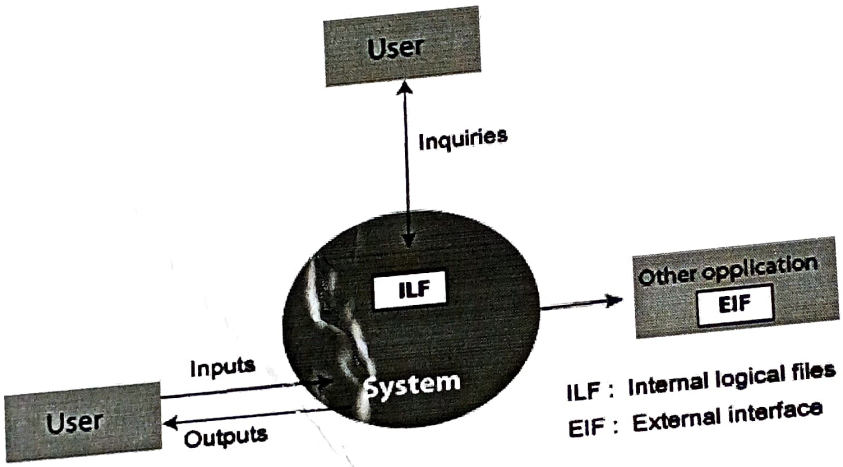
Measurements Parameters	Examples
.Number of External Inputs(EI)	Input screen and tables
. Number of External Output (EO)	Output screens and reports

Number of external inquiries (EQ)	Prompts and interrupts.
Number of internal files (ILF)	Databases and directories
Number of external interfaces (EIF)	Shared databases and shared rou

Types of FP Attributes

All these parameters are then individually assessed for complexity.

The FPA functional units are shown in Fig:



FPAs Functional Units System

2. FP characterizes the complexity of the software system and hence can be used to depict the project time and the manpower requirement.

3. The effort required to develop the project depends on what the software does.

4. FP is programming language independent.

5. FP method is used for data processing systems, business systems like information systems.

6. The five parameters mentioned above are also known as information domain characteristics.

7. All the parameters mentioned above are assigned some weights that have been experimentally determined and are shown in Table

Weights of 5-FP Attributes

Measurement Parameter	Low	
. Number of external inputs (EI)	7	1
. Number of external outputs (EO)	5	7
. Number of external inquiries (EQ)	3	4
. Number of internal files (ILF)	4	5
. Number of external interfaces (EIF)	3	4

The functional complexities are multiplied with the corresponding weights against each function, and the values are added up to determine the UFP (Unadjusted Function Point) of the subsystem.

Computing FPs

Measurement Parameter	Count		Weighing factor			
			Simple	Average	Complex	
1. Number of external inputs (EI)	—	*	3	4	6 =	—
2. Number of external Output (EO)	—	*	4	5	7 =	—
3. Number of external inquiries (EQ)	—	*	3	4	6 =	—
4. Number of internal Files (ILF)	—	*	7	10	15 =	—
5. Number of external interfaces(EIF)	—	*	5	7	10 =	—
Count-total →						

Here that weighing factor will be simple, average, or complex for a measurement parameter type.

The Function Point (FP) is thus calculated with the following formula.

$$FP = \text{Count-total} * [0.65 + 0.01 * \sum(f_i)]$$
$$= \text{Count-total} * CAF$$

where Count-total is obtained from the above Table.

$$\mathbf{CAF = [0.65 + 0.01 * \sum(f_i)]}$$

and $\mathbf{\sum(f_i)}$ is the sum of all 14 questionnaires and show the complexity adjustment value/ factor-CAF (where i ranges from 1 to 14). Usually, a student is provided with the value of $\sum(f_i)$

Also note that $\mathbf{\sum(f_i)}$ ranges from 0 to 70, i.e.,

$$0 <= \sum(f_i) <= 70$$

and CAF ranges from 0.65 to 1.35 because

a. When $\mathbf{\sum(f_i)}$ = 0 then CAF = 0.65

b. When $\mathbf{\sum(f_i)}$ = 70 then CAF = 0.65 + (0.01 * 70) = 0.65 + 0.7 = 1.35

Based on the FP measure of software many other metrics can be computed:

a. Errors/FP

b. $/FP.

c. Defects/FP

d. Pages of documentation/FP

e. Errors/PM.

f. Productivity = FP/PM (effort is measured in person-months).

g. $/Page of Documentation.

8. LOCs of an application can be estimated from FPs. That is, they are interconvertible. **This process is known as backfiring**. For example, 1 FP is equal to about 100 lines of COBOL code.

9. FP metrics is used mostly for measuring the size of Management Information System (MIS) software.

10. But the function points obtained above are unadjusted function points (UFPs). These (UFPs) of a subsystem are further adjusted by considering some more General System Characteristics (GSCs). It is a set of 14 GSCs that need to be considered. The procedure for adjusting UFPs is as follows:

Output:

a. Degree of Influence (DI) for each of these 14 GSCs is assessed on a scale of 0 to 5. (b) If a particular GSC has no influence, then its weight is taken as 0 and if it has a strong influence then its weight is 5.

b. The score of all 14 GSCs is totaled to determine Total Degree of Influence (TDI).

c. Then Value Adjustment Factor (VAF) is computed from TDI by using the formula: **VAF = (TDI * 0.01) + 0.65**

Remember that the value of VAF lies within 0.65 to 1.35 because

a. When TDI = 0, VAF = 0.65

b. When TDI = 70, VAF = 1.35

c. VAF is then multiplied with the UFP to get the final FP count: **FP = VAF * UFP**

Example: Compute the function point, productivity, documentation, cost per function for the following data:

1. Number of user inputs = 24
2. Number of user outputs = 46
3. Number of inquiries = 8
4. Number of files = 4
5. Number of external interfaces = 2
6. Effort = 36.9 p-m
7. Technical documents = 265 pages
8. User documents = 122 pages
9. Cost = $7744/ month

Various processing complexity factors are: 4, 1, 0, 3, 3, 5, 4, 4, 3, 3, 2, 2, 4, 5.

Measurement Parameter	Count
. Number of external inputs (EI)	24
. Number of external outputs (EO)	46
. Number of external inquiries (EQ)	8
. Number of internal files (ILF)	4

.umber of external interfaces (EIF) Count-total →	2

Solution:

So sum of all f_i (i ← 1 to 14) = 4 + 1 + 0 + 3 + 5 + 4 + 4 + 3 + 3 + 2 + 2 + 4 + 5 = 43

$$FP = \text{Count-total} * [0.65 + 0.01 * \sum(f_i)]$$
$$= 378 * [0.65 + 0.01 * 43]$$
$$= 378 * [0.65 + 0.43]$$

$$= 378 * 1.08 = 408$$

$$\text{Productivity} = \frac{FP}{\text{Effort}} = \frac{408}{36.9} = 11.1$$

Total pages of documentation = technical document + user document
= 265 + 122 = 387pages

Documentation = Pages of documentation/FP
= 387/408 = 0.94

$$\text{Cost per function} = \frac{\text{cost}}{\text{productivity}} = \frac{7744}{11.1} = \$700$$

Differentiate between FP and LOC

FP	LOC
. FP is specification based.	1. LOC is an analogy based.
. FP is language independent.	2. LOC is language dependent.
. FP is user-oriented.	3. LOC is design-oriented.
. It is extendible to LOC.	4. It is convertible to FP (backfiring)