❖ Project Estimation

Software measurement is an indicator of size, quantity, amount or dimension of a particular attribute of a product or process.

It helps the project manager and entire software team to make decisions that lead to successful completion of the project by generating quantity results.

Two categories of software measurements:

- Direct Measures: It includes software processes like Cost and Effort applied, Lines of code produced, Execution speed and total no. of errors that have been reported.
- Indirect Measures: It includes products like functionality, quality, complexity, reliability, maintainability and many more.

Software Metrics

Software metrics provide measures, functions or formulas for various aspects of software process and product.

It including measuring software performance, planning work items, measuring productivity and many other uses

Software metrics of three categories:

- Product metrics: it estimates size, complexity, quality and reliability of software.
- Process metrics: it estimates fault rate during development, pattern of testing defect arrival, time it takes for a fixed operation.
- Project metrics: it estimates the number of software developers, cost, scheduling and productivity of software.

Size metrics

Size oriented metrics concentrates on the size of the program created.

- LOC(Lines of Code)

- FP(Functional Point)

❖ LOC

A line of code (LOC) is any line of text in a code that is not a comment or blank line, and also header lines, in any case of the number of statements or fragments of statements on the line. LOC clearly consists of all lines containing the declaration of any variable, and executable and non-executable statements. As Lines of Code (LOC) only counts the volume of code, you can only use it to compare or estimate projects that use the same language and are coded using the same coding standards.

Features of Lines of Code (LOC):

- Change Tracking: Variations in LOC as time passes can be tracked to analyze the growth or reduction of a codebase, providing insights into project progress.
- Limited Representation of Complexity: Despite LOC providing a general idea of code size, it does not accurately depict code complexity. It is possible for two programs having the same LOC to be incredibly complex.
- Ease of Computation: LOC is an easy measure to obtain because it is easy to calculate and takes little time.
- Easy to Understand: The idea of expressing code size in terms of lines is one that stakeholders, even those who are not technically inclined, can easily understand.

Advantages of Lines of Code (LOC):

- Effort Estimation: LOC is occasionally used to estimate development efforts and project deadlines at a high level. Although caution is necessary, project planning can begin with this.

- Comparative Analysis: High-level productivity comparisons between several projects or development teams can be made using LOC. It might provide an approximate figure of the volume of code generated over a specific time frame.
- Benchmarking Tool: When comparing various iterations of the same program, LOC can be used as a benchmarking tool. It may bring information on how modifications affect the codebase's total size.

Disadvantages of Lines of Code (LOC):

- Challenges in Agile Work Environments: Focusing on initial LOC estimates may not adequately reflect the iterative and dynamic nature of development in agile development, as requirements may change.
- Not Considering Into Account External Libraries: Code from other libraries or frameworks, which can greatly enhance a project's overall usefulness, is not taken into account by LOC.
- Challenges with Maintenance: Higher LOC codebases are larger codebases that typically demand more maintenance work.

❖ FP

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 (Function Point Analysis) is used to make an 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.

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.

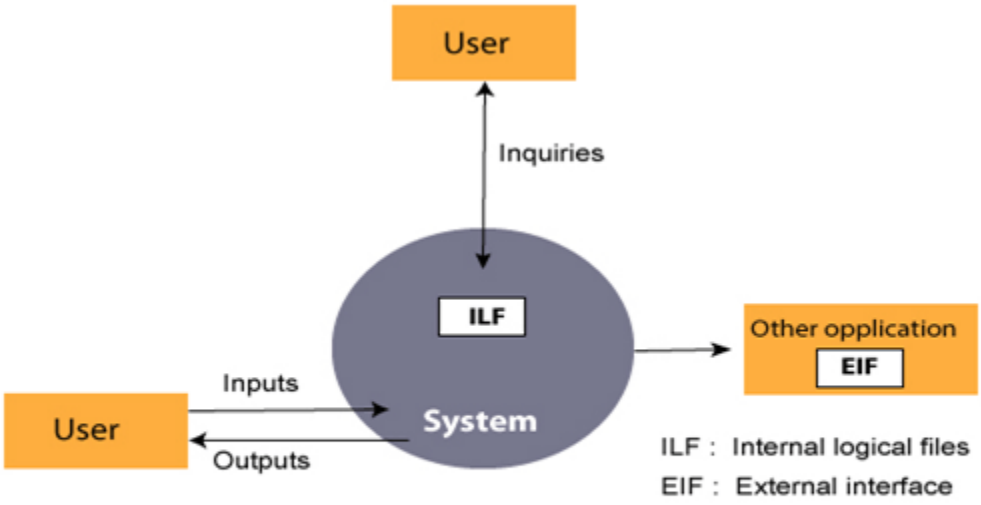
Following are the points regarding FPs

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**
1.Number of External Inputs(EI)	Input screen and tables
2. Number of External Output (EO)	Output screens and reports
3. Number of external inquiries (EQ)	Prompts and interrupts.
4. Number of internal files (ILF)	Databases and directories
5. Number of external interfaces (EIF)	Shared databases and shared routines.

All these parameters are then individually assessed for complexity.

The FPA functional units are shown in Fig:



FPAs Functional Units System

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

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

3. FP is programming language independent.

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

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

6. 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	Average	High
1. Number of external inputs (EI)	7	10	15
2. Number of external outputs (EO)	5	7	10
3. Number of external inquiries (EQ)	3	4	6
4. Number of internal files (ILF)	4	5	7
5. Number of external interfaces (EIF)	3	4	6

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 = Count-total * [0.65 + 0.01 * ∑(fi)]

= Count-total * CAF

where ,

Count-total is obtained from the above Table.

CAF = [0.65 + 0.01 *∑(fi)]

and ∑(fi) 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 ∑(fi)

Also note that ∑(fi) ranges from 0 to 70, i.e., 0 <= ∑(fi) <=70

7. 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.

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

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

Solutions:

Measurement Parameter	Count		Weighing factor
1. Number of external inputs (EI)	24	*	4 = 96
2. Number of external outputs (EO)	46	*	4 = 184
3. Number of external inquiries (EQ)	8	*	6 = 48
4. Number of internal files (ILF)	4	*	10 = 40
5. Number of external interfaces (EIF) Count-total	2	*	5 = 10
Total			378

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

FP = Count-total * [0.65 + 0.01 *∑(fi)]

= 378 * [0.65 + 0.01 * 43]

= 378 * [0.65 + 0.43]

= 378 * 1.08 = 408