Practical 3

Function Point Analysis and COCOMO Model

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 an estimate of the software project, including its testing in terms of functionality or function size of the software product.

The Functional Point Analysis (FPA) is a one of the most popularly used software estimation technique to measure the functional size of the software work i.e. It is a method or set rules to measure the amount of software functionalities and software size of the software developed product. It depends on logical view flow of the application i.e. the number of functionalities delivered to the users. The Functional Point (FP) is the measurement of functional size of the software application.

When a new project comes to organisation, the organisation starts planning and estimation on that project in terms of schedule time, cost, resources, etc. The estimation process of software product includes the testing phase in terms of every functionality of the application, so the name shows functional point analysis.

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

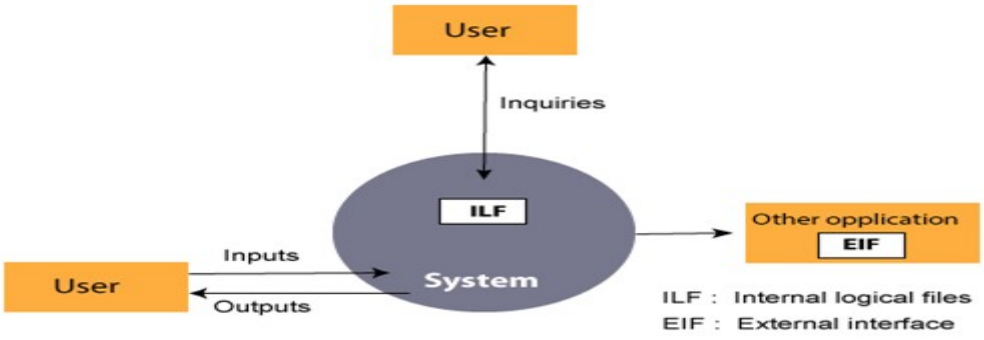
1. FPs of an application is found out by counting the number and types of functions used in the applications. Various functions used in an application can be put under five types, as shown in Table:

Types of FP Attributes

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

(a)An **Internal Logical File (ILF)** is a **user identifiable group of logically related data or control information maintained within the application scope**. The **primary purpose** of an ILF is to hold data maintained through one or more elementary processes of the primary application being counted. For example, A pilot may feed navigation data via a display before journey. Data is stored in a file & can be modified in future. Hence, the pilot is responsible for maintaining the file containing navigational information.
(b) An **External Interface File (EIF)** is a **user identifiable group of logically related data or control information referenced by the application, but maintained within the scope** of another application(ILF). The **primary purpose** of an EIF is to hold data transferred through one or more elementary processes within the boundary of the application counted. For example; It may be necessary for the pilot to reference position data from satellite or ground based facility during the flight. Here the pilot isn't responsible for updating data on these sites but must reference it during the journey.
(c)An**External Input (EI)** is the **smallest unit of activity that is meaningful to the end user in the business**. The **primary purpose** of an EI is to maintain one or more ILF and/or to alter the behaviour of the system. For example, the pilot can add, change and delete navigational information before or during the journey. In a way, he/she is utilizing a transaction or EI which gives the capability to maintain data through adding, changing & deleting it's content.
(d) The **External Output (EO)** is an elementary process that **generates data or control information sent outside the application's boundary.** The **primary purpose** of an EO is to present information to user through processing logic other than or in addition to the retrieval of data or control information. For example, the pilot has the ability to display ground speed, true air speed & calibrated air speed. The results displayed here is derived from the maintained & referenced data or EO.
(e) An **External Enquiry (EQ)**is an elementary process made up of an **input-output combination that results in data retrieval**. The **primary purpose** of an EQ is to present information to the user through the retrieval of data or control information. For example, the pilot displays terrain clearance data that was previously set, the resulting output is direct retrieval of stored information (EQ).

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	Average	High
1. Number of external inputs (EI)	3	4	6
2. Number of external outputs (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

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 →						

Function Point (FP) is an element of software development which helps to approximate the cost of development early in the process. It may measure functionality from the user's point of view.

Counting Function Point (FP):

- **Step-1:**

  F = 14 * scale

  Scale varies from 0 to 5 according to character of Complexity Adjustment Factor (CAF). Below table shows scale:

  0 - No Influence

  1 - Incidental

  2 - Moderate

  3 - Average

4 - Significant
5 - Essential

14 aspects of processing complexity questions are as follow:

General System Characteristic	Brief Description
Data Communications	How many communication facilities are there to aid in the transfer or exchange of information with the application or system?
Distributed Data Processing	How are distributed data and processing functions handled?
Performance	Did the user require response time or throughput?
Heavily Used Configuration	How heavily used is the current hardware platform where the application will be executed?
Transaction Rate	How frequently are transactions executed daily, weekly, monthly, etc.?
On-Line Data Entry	What percentage of the information is entered online?
End-user Efficiency	Was the application designed for end-user efficiency?
Online Update	How many ILFs are updated by online transaction?
Complex Processing	Does the application have extensive logical or mathematical processing?
Reusability	Was the application developed to meet one or many user's needs?
Installation Ease	How difficult is conversion and installation?
Operational Ease	How effective and/or automated are start-up, back-up, and recovery procedures?
Multiple Sites	Was the application specifically designed, developed, and supported to be installed at multiple sites for multiple organizations?
Facilitate Change	Was the application specifically designed, developed, and supported to facilitate change?

General System Characteristic Glossary (Credits: platformrenderapps.com)

- 0 = No Influence
- 1 = Incidental
- 2 = Moderate
- 3 = Average
- 4 = Significant
- 5 = Essential

General System Characteristics (GSCs)	Degree of Influence (DI) 0 - 5
1. Data Communications	_____
2. Distributed Data Processing	_____
3. Performance	_____
4. Heavily Used Configuration	_____
5. Transaction Rate	_____
6. Online Data Entry	_____
7. End-User Efficiency	_____
8. Online Update	_____
9. Complex Processing	_____
10. Reusability	_____
11. Installation Ease	_____
12. Operational Ease	_____
13. Multiple Sites	_____
14. Facilitate Change	_____
Total Degree of Influence (TDI)	_____
Value Adjustment Factor (VAF)	_____
	VAF = (TDI * 0.01) + 0.65

- **Step-2:** Calculate Complexity Adjustment Factor (CAF).
  CAF = 0.65 + (0.01 * F) or VAF = (TDI * 0.01) + 0.65
- **Step-3:** Calculate Unadjusted Function Point (UFP).
  TABLE (Required)

Function Units	Low	Avg	High
EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EIF	5	7	10

  Multiply each individual function point to corresponding values in TABLE.
- **Step-4:** Calculate Function Point.
  FP = UFP * CAF

Example:
Given the following values, compute function points when all complexity adjustment factors (CAF) and weighting factors are average.
User Input = 50
User Output = 40
User Inquiries = 35
User Files = 6
External Interface = 4
Explanation:

- **Step-1:** As complexity adjustment factor is average (given in question), hence,
- scale = 3.

F = 14 * 3 = 42

- **Step-2:**
  CAF = 0.65 + (0.01 * 42) = 1.07
- **Step-3:** As weighting factors are also average (given in question) hence we will multiply each individual function point to corresponding values in TABLE.
  UFP = (50*4) + (40*5) + (35*4) + (6*10) + (4*7) = 628
- **Step-4:**
  Function Point = 628 * 1.07 = 671.96
  This is the required answer.

COCOMO Model

COCOMO (Constructive Cost Estimation Model) model was proposed by Boehm (1981). According to Boehm, software cost estimation should be done through three stages: Basic COCOMO, Intermediate COCOMO, and Complete COCOMO.

The necessary steps in this model are:

1. Get an initial estimate of the development effort from evaluation of thousands of delivered lines of source code (KDLOC).
2. Determine a set of 15 multiplying factors from various attributes of the project.
3. Calculate the effort estimate by multiplying the initial estimate with all the multiplying factors i.e., multiply the values in step1 and step2.

The initial estimate (also called nominal estimate) is determined by an equation of the form used in the static single variable models, using KDLOC as the measure of the size. To determine the initial effort E_i in person-months the equation used is of the type is shown below

$$\mathbf{E_i = a*(KDLOC)^b}$$

The value of the constant a and b are depends on the project type.

In COCOMO, projects are categorized into three types:Organic, Semidetached and Embedded Software Projects

Organic: A development project can be considered of organic type, if the project deals with developing a well understood application program, the size of the development team is reasonably small, and the team members are experienced in developing similar types of projects.**Examples of this type of projects are simple business systems, simple inventory management systems, and data processing systems.**

Semi-detached: A development project can be considered of semi-detached type, if the development consists of a mixture of experienced and inexperienced staff. Team members may have limited experience on related systems but may be unfamiliar with some aspects of the system being developed.**Example of Semidetached system includes developing a new operating system (OS), a Database Management System (DBMS), and complex inventory management system.**

Embedded: A development project is considered to be of embedded type, if the software being developed is strongly coupled to complex hardware, or if the stringent regulations on the operational procedures exist.**For Example:** ATM, Air Traffic control.

Three stages of software cost estimation: Basic, Intermediate, Detail Model

Basic COCOMO Model

The **basic COCOMO model** gives an approximate estimate of the project parameters. The basic COCOMO estimation model is given by the following expressions:

$$\text{Effort} = a_1 \times (\text{KLOC})^{a_2}\ \text{PM}$$

$$\text{Tdev} = b_1 \times (\text{Effort})^{b_2}\ \text{Months}$$

where,

- **KLOC** is the estimated size of the software product expressed in Kilo Lines of Code,
- **a1**, **a2**, **b1**, **b2** are constants for each category of software products,
- **Tdev** is the estimated time to develop the software, expressed in months,
- **Effort** is the total effort required to develop the software product, expressed in person months (PMs).

The values of a1, a2, b1, b2 for different categories of products (i.e., organic, semi-detached, and embedded) as given by Boehm are summarized below.

Estimation of Development Effort: For the three classes of software products, the formulas for estimating the effort based on the code size are shown below:

Organic: $\text{Effort} = 2.4 \times (\text{KLOC})^{1.05}\ \text{PM}$

Semi-detached: $\text{Effort} = 3.0 \times (\text{KLOC})^{1.12}\ \text{PM}$

Embedded: $\text{Effort} = 3.6 \times (\text{KLOC})^{1.20}\ \text{PM}$

Estimation of Development Time: For the three classes of software products, the formulas for estimating the development time based on the effort size are given below:

Organic: $\text{Tdev} = 2.5 \times (\text{Effort})^{0.38}\ \text{Months}$

Semi-detached: $\text{Tdev} = 2.5 \times (\text{Effort})^{0.35}\ \text{Months}$

Embedded: $\text{Tdev} = 2.5 \times (\text{Effort})^{0.32}\ \text{Months}$

Example: Assume that the size of an organic type software product has been estimated to be 32,000 lines of source code. Assume that the average salary of software engineers be Rs. 15,000/- per month. Determine the effort required to develop the software product, the nominal development time, and cost required to develop the product. Also find number of persons required.

From the basic COCOMO estimation formula for organic software:

$$\text{Effort} = 2.4 \times (32)^{1.05} = 91\ \text{PM}$$

$$\text{Tdev} = 2.5 \times (91)^{0.38} = 14\ \text{Months}$$

$$\text{Cost} = 14 \times 15{,}000 = \text{Rs.}\ 2{,}10{,}000/-$$

Person required (P) = Effort / Tdev = 91/14=6.5 Person ~7Person

The key parameters which define the quality of any software products, which are also an outcome of the Cocomo are primarily Effort & Schedule:

- **Effort:** Amount of labor that will be required to complete a task. It is measured in person-months units.
- **Schedule:** Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put in. It is measured in the units of time such as weeks, and months

Lab Work:

- Perform Estimation using Function Point Analysis and Basic COCOMO model for your system. [Make proper assumptions.]