Function Point (FP)

Definition: It's a component of software development which helps to approximate the cost of development early in the process. It is a process which defines the required functions and their complexity in a piece of software in order to estimate the software's size and scope upon completion.

To compute the number of FPs we first compute an unadjusted function point count (UFC).

$$\text{UFC} = \sum_{i=1}^{15} (\text{Number of items of variety } i) \times (\text{weight}_i)$$

1. **External input (EI)**
   1: admin and student registered their account
   2: Admin and student user view their dashboard
   3: Admin and student can edit their profile
   4: Admin add books
   5: Admin delete books
   6: Admin issue new book and update details
2. **External output (EO)**
   1: Student view issue book and return date
   2: Admin can approve book request
   3: Admin can search student
   4: Admin charge fine send notification to give fine
3. **External inquiries (EIq)**
   1: Student send book request by book's ISBN number
   2: Admin can see feedback R12: Student can give feedback and also see other's feedback
4. **External Files (EF)**
   1:Student see their expired date and notified fine
   2: Admin and student can recover their password R16: Admin and student can see book list
   3: Admin can see student's details

5. **Internal Files (IF)**

1: Database file

2: Root Direectory

Function Point Complexity Weights

Item	Simple	Weighting Factor Average	Complex
External inputs	3	4	6
External outputs	4	5	7
External inquiries	3	4	6
External files	7	10	15
Internal files	5	7	10

calculated in java

Components of the Technical Complexity Factor

F_1 Reliable backup and recovery	F_2 Data communications
F_3 Distributed functions	F_4 Performance
F_5 Heavily used configuration	F_6 Online data entry
F_7 Operational ease	F_8 Online update
F_9 Complex interface	F_{10} Complex processing
F_{11} Reusability	F_{12} Installation ease
F_{13} Multiple sites	F_{14} Facilitate change

Degree of Significance

0	Irrelevant
1	Insignificant
2	Moderate
3	Average
4	Significant
5	Essential

ID	Technical complexity Factor	Complexity
F1	Reliable backup and recovery	**4**
F2	Data communications	**2**
F3	Distributed functions	**3**
F4	Performance	**0**
F5	Heavily used configuration	**2**
F6	Online data entry	**5**
F7	Operational ease	**1**
F8	Online update	**5**
F9	Complex interface	**4**
F10	Complex processing	**5**
F11	Reusability	**3**
F12	Installation ease	**0**
F13	Multiple sites	**7**
F14	Facilitate change	**3**

Technical Complexity Factor

$$TCF = 0.65 + 0.01 \sum_{i=1}^{14} F_i$$

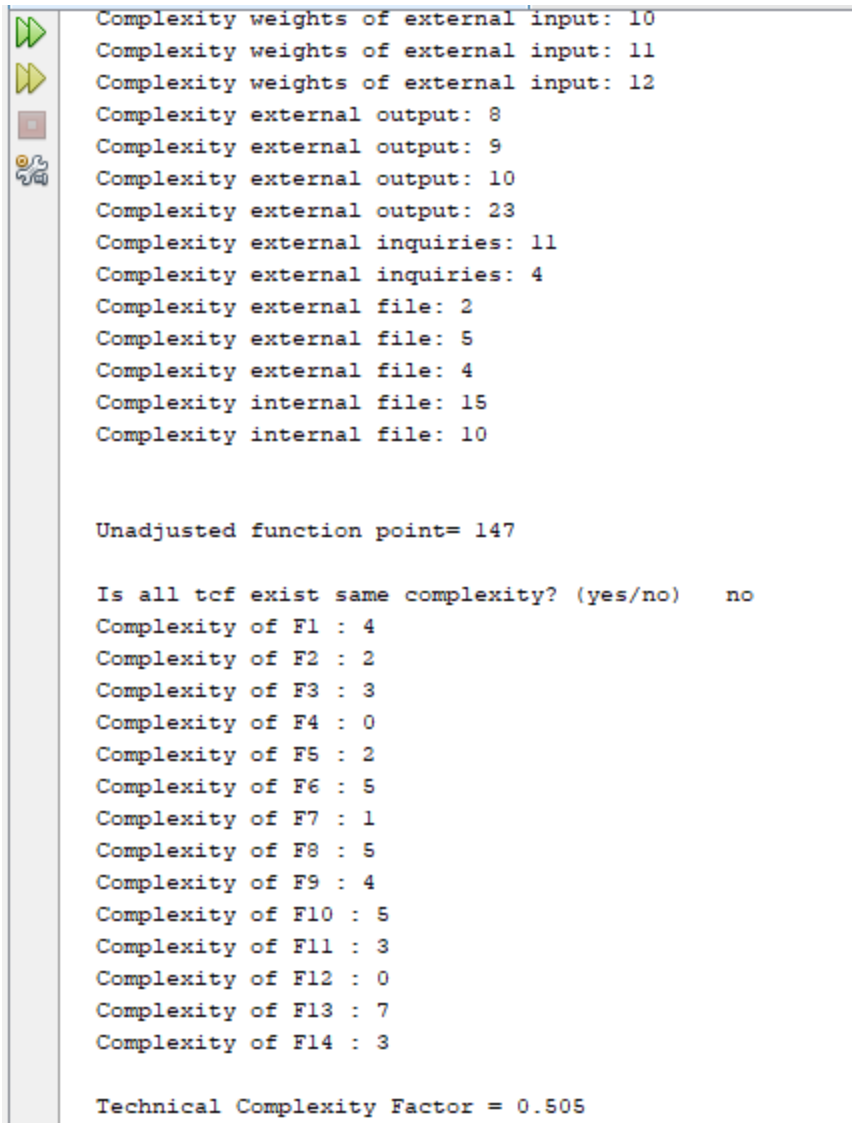
Functional Point

$$FP = UFC \times TCF$$

- Measure techniques : Manually / programatic
- Requirement : SRS

```
package software_metrics_lab_final;

import java.util.Scanner;


public class Function_Point {
     public static void main(String arg[])
    {
        Scanner sc = new Scanner(System.in);

        System.out.print("External Input: ");
        int ei = sc.nextInt();

        System.out.print("External Output: ");
        int eo = sc.nextInt();

        System.out.print("External Inquiries: ");
        int eiq = sc.nextInt();

        System.out.print("External files: ");
        int ef = sc.nextInt();

        System.out.print("Internal files: ");
        int If = sc.nextInt();



        int sum = 0;
          for (int i = 0; i < ei; i++)
        {
        System.out.print("Complexity weights of external input: ");
        int c = sc.nextInt();
        sum = sum + c;
```

```
35
36
37              for (int j = 0; j < eo; j++)
38          {
39          System.out.print("Complexity external output: ");
40          int c = sc.nextInt();
41          sum = sum + c;
42          }
43
44             for (int k = 0; k < eiq; k++)
45          {
46          System.out.print("Complexity external inquiries: ");
47          int c = sc.nextInt();
48          sum = sum + c;
49          }
50
51             for (int l = 0; l < ef; l++)
52          {
53          System.out.print("Complexity external file: ");
54          int c = sc.nextInt();
55          sum = sum + c;
56          }
57
58             for (int m = 0; m < If; m++)
59          {
60          System.out.print("Complexity internal file: ");
61          int c = sc.nextInt();
62          sum = sum + c;
63          }
```



```
Complexity weights of external input: 10
Complexity weights of external input: 11
Complexity weights of external input: 12
Complexity external output: 8
Complexity external output: 9
Complexity external output: 10
Complexity external output: 23
Complexity external inquiries: 11
Complexity external inquiries: 4
Complexity external file: 2
Complexity external file: 5
Complexity external file: 4
Complexity internal file: 15
Complexity internal file: 10


Unadjusted function point= 147

Is all tcf exist same complexity? (yes/no)   no
Complexity of F1 : 4
Complexity of F2 : 2
Complexity of F3 : 3
Complexity of F4 : 0
Complexity of F5 : 2
Complexity of F6 : 5
Complexity of F7 : 1
Complexity of F8 : 5
Complexity of F9 : 4
Complexity of F10 : 5
Complexity of F11 : 3
Complexity of F12 : 0
Complexity of F13 : 7
Complexity of F14 : 3

Technical Complexity Factor = 0.505
```

```
67          System.out.println("Unadjusted function point= "+ufc);
68
            int cf1=0,cf,sum1=0;
70          double tcf;
71
72          System.out.print("\nIs all tcf exist same complexity? (yes/no)    ");
73          String q = sc.next();
74
75          if(q.equals("yes")){
76          System.out.print("Complexity of TCF: ");
77          cf = sc.nextInt();
78          tcf = 0.065 + 0.01 * (14*cf);
79          }
80          else{
81          for (int i=1; i<=14; i++)
82             {
83              System.out.print("Complexity of F"+i+" : ");
84              cf1 = sc.nextInt();
85              sum1=sum1+cf1;
86             }
87            cf = sum1;
88            tcf = 0.065 + 0.01 * cf;
89          }
90
91          System.out.println("\nTechnical Complexity Factor = "+tcf);
92
93          System.out.println("\n");
94          double fP;
95          fP = ufc * tcf;
96          System.out.println("Function Point = "+fP);
97          System.out.println("\n\n");
98
99      }
00  }
```