**Experiment No. 7: Estimate the Cost of Your Project**

### **Aim:**

To estimate the cost of the software project using appropriate cost estimation techniques and models.

### **Theory:**

Cost estimation in software engineering is the process of predicting the resources, effort, time, and money required to develop a software project. Accurate cost estimation helps in planning, budgeting, and resource allocation, ensuring the success of the project.

**Common Cost Estimation Techniques:**

1. **COCOMO (Constructive Cost Model):** A mathematical model that estimates cost based on the size of the project in terms of lines of code (LOC).
2. **Function Point Analysis (FPA):** Focuses on functional requirements and quantifies them for cost estimation.
3. **Expert Judgment:** Relies on the experience of experts to provide cost predictions.
4. **Analogy-Based Estimation:** Compares the current project with similar past projects to estimate costs.
5. **Parametric Models:** Uses parameters like effort, schedule, and quality to calculate costs.

**Factors Influencing Cost Estimation:**

* Project size
* Team experience and expertise
* Development model chosen
* Project complexity
* Resource availability

The function point method was originally developed by Bij Albrecht. A function point is a rough estimate of a unit of delivered functionality of a software project. Function points (FP) measure size in terms of the amount of functionality in a system. Function points are computed by first calculating an unadjusted function point count (UFC). Counts are made for the following categories

• Number of user inputs

Each user input that provides distinct application oriented data to the software is counted.

• Number of user outputs

Each user output that provides application oriented information to the user is counted. In this context "output" refers to reports, screens, error messages, etc. Individual data items within a report are not counted separately.

• Number of user inquiries

An inquiry is defined as an on-line input that results in the generation of some immediate software response in the form of an on-line output. Each distinct inquiry is counted.

• Number of files

Each logical master file is counted.

• Number of external interfaces

All machine-readable interfaces that are used to transmit information to another system are counted.

Once this data has been collected, a complexity rating is associated with each count according to Table

TABLE 1: Function point complexity weights.

|  |  |  |  |
| --- | --- | --- | --- |
| Measurement parameter | Weighting factor | | |
|  | Simple | Average | Complex |
| Number of user inputs | 3 | 4 | 6 |
| Number of user outputs | 4 | 5 | 7 |
| Number of user inquiries | 3 | 4 | 6 |
| Number of files | 7 | 10 | 15 |
| Number of external interfaces | 5 | 7 | 10 |

Each count is multiplied by its corresponding complexity weight and the results are summed to provide the UFC. The adjusted function point count (FP) is calculated by multiplying the UFC by a technical complexity factor (TCF) also referred to as Value Adjustment Factor (VAF). Components of the TCF are listed in Table 2

Table 2. Components of the technical complexity factor.

|  |  |  |  |
| --- | --- | --- | --- |
| F1 | Reliable back-up and recovery | F2 | Data communications |
| F3 | Distributed functions | F4 | Performance |
| F5 | Heavily used configuration | F6 | Online data entry |
| F7 | Operational ease | F8 | Online update |
| F9 | Complex interface | F10 | Complex processing |
| F11 | Reusability | F12 | Installation ease |
| F13 | Multiple sites | F14 | Facilitate change |

Alternatively, the following questionnaire could be utilized

1. Does the system require reliable backup and recovery?

2. Are data communications required?

3. Are there distributed processing functions?

4. Is performance critical?

5. Will the system run in an existing, heavily utilized operational environment?

6. Does the system require on-line data entry?

7. Does the on-line data entry require the input transaction to be built over multiple screens or operations?

8. Are the master files updated online?

9. Are the input, outputs, files or inquiries complex?

10. Is the internal processing complex?

11. Is the code designed to be reusable?

12. Are conversions and installation included in the design?

13. Is the system designed for multiple installations in different organizations?

14. Is the applications designed to facilitate change and ease of use?

Each component is rated from 0 to 5, where 0 means the component has no influence on the system and 5 means the component is essential (Pressman, 1997). The VAF can then be calculated as:

VAF = 0.65 + (Sum of GSCs x 0.01) Where Sum of GSCs = SUM(Fi)

The factor varies from 0.65 (if each Fi is set to 0) to 1.35 (if each Fi is set to 5) (Fenton, 1997). The final function point calculation is:

Final Adjusted FP = UFC x VAF

Convert AFP into SLOC using appropriate conversion factor.

SLOC = 16 x SLOC/AFP [NOTE: 16 is the conversion factor]

EFFORT = EAF x A x (SLOC)EX

EAF = CPLX x TOOL

A = 3.2= Constant based on the development mode.

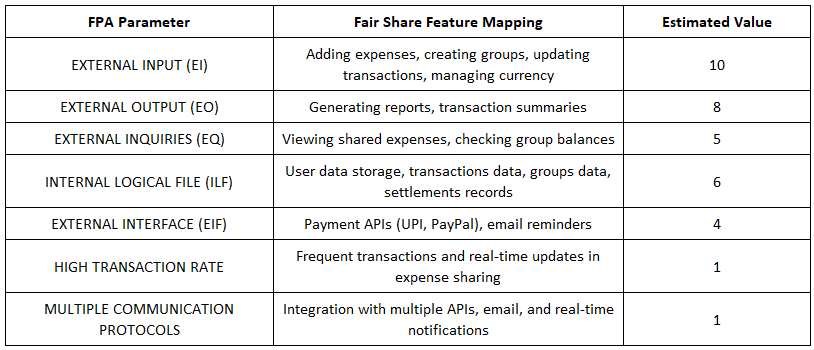
EX = 0.38= Constant based on the development mode.

CPLX = 1.3 = Constant based on the development language.

TOOL = 1.1 = Constant based on the development Tool.

TDEV = 2.5 x (EFFORT) EX in months.

**FPA Model:**



PROGRAM CODE:

class FunctionPointEstimation:

    def \_\_init\_\_(self):

        self.weights = [[3, 4, 5], [4, 5, 7], [3, 4, 6], [7, 10, 15], [5, 7, 10]]

    def menu(self):

        print("1. Low\n2. Average\n3. High")

        return int(input("Enter choice: "))

    def wfactor(self, r, h):

        return self.weights[r - 1][h - 1]

    def calculate\_fp(self):

        sum\_fp = 0

        params = ["external input", "external output", "external inquiries",

                  "internal logical file", "external interface"]

        for i, param in enumerate(params):

            value = int(input(f"Enter {param} value: "))

            sum\_fp += value \* self.wfactor(i+1, self.menu())

        high\_transaction\_rate = int(input("Enter high transaction rate: "))

        multiple\_comm\_protocol = int(input("Enter multiple communication protocols: "))

        print(f"UFP = {sum\_fp}")

        TDI = high\_transaction\_rate + multiple\_comm\_protocol

        VAF = 0.65 + (TDI \* 0.01)

        AFP = sum\_fp \* VAF

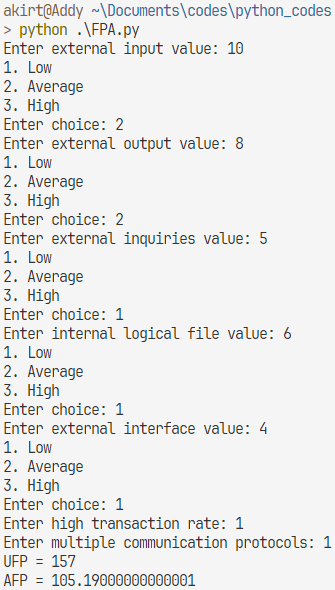
        print(f"AFP = {AFP}")

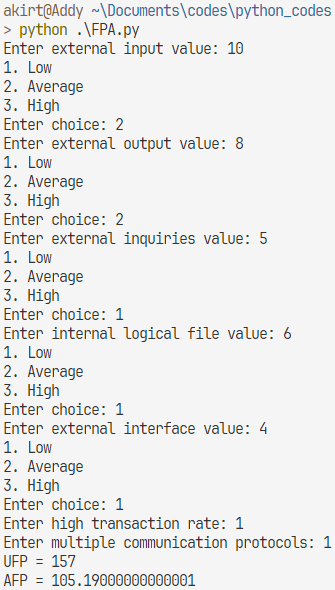
if \_\_name\_\_ == "\_\_main\_\_":

    fp\_model = FunctionPointEstimation()

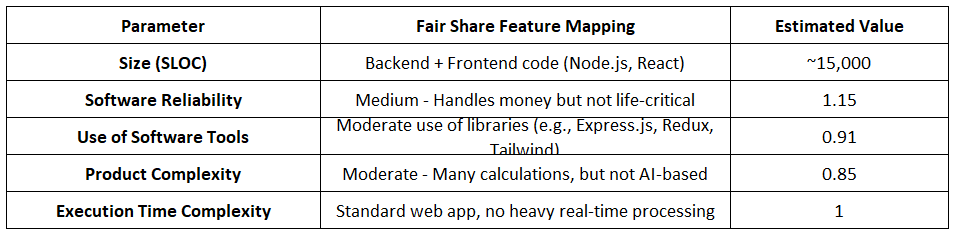
    fp\_model.calculate\_fp()

**RESULT/ OUTPUT:**





**COCOMO Model:**



PROGRAM CODE:

import math

class COCOMOModel:

    def \_\_init\_\_(self):

        self.a\_values = {"organic": 3.2, "semidetached": 3.0, "embedded": 2.8}

        self.b\_values = {"organic": 1.05,

                         "semidetached": 1.12, "embedded": 1.20}

    def calculate\_effort(self, size, EAF, model\_type):

        a = self.a\_values[model\_type]

        b = self.b\_values[model\_type]

        return a \* (size \*\* b) \* EAF

    def main(self):

        size = float(input("Enter Size (KLOC): "))

        reliability = float(input("Enter Software Reliability: "))

        software\_tools = float(input("Enter Use of Software Tools: "))

        product\_complexity = float(input("Enter Product Complexity: "))

        execution\_complexity = float(

            input("Enter Execution Time Complexity: "))

        EAF = reliability \* software\_tools \* product\_complexity \* execution\_complexity

        for model\_type in ["organic", "semidetached", "embedded"]:

            effort = self.calculate\_effort(size, EAF, model\_type)

            print(

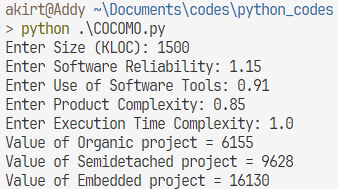
                f"Value of {model\_type.capitalize()} project = {math.ceil(effort)}")

if \_\_name\_\_ == "\_\_main\_\_":

    cocomo = COCOMOModel()

    cocomo.main()

### **RESULT /OUTPUT:**



### **Learning Objective:**

* To understand various cost estimation techniques and their applications.
* To learn how to estimate project costs systematically.
* To develop skills for budget planning in software projects.

### **Learning Outcome:**

At the end of this experiment, students will be able to:

1. Analyze the project requirements to determine cost estimation inputs.
2. Apply cost estimation techniques to predict project costs accurately.
3. Evaluate the impact of resource allocation and project complexity on cost estimation.

### **Course Outcomes (COs)**

* **CO3:** Identify and formulate functional and non-functional requirements for software systems and perform cost estimation using various models.
* **CO5:** Evaluate and mitigate software project risks and apply configuration management practices to maintain project integrity.

### **Cognitive Levels of Attainment as per Bloom’s Taxonomy**

* **L3 (Apply):** Apply cost estimation techniques to the project.
* **L5 (Evaluate):** Evaluate the accuracy and feasibility of cost estimation.

### **Programme Outcome (POs):**

* **PO2: Problem Analysis:** Analyze project requirements to identify cost-related factors.
* **PO3: Design/Development of Solutions:** Design and calculate cost estimation models for the project.
* **PO9: Communication**: Document and present cost estimation details effectively.
* **PO10: Project Management and Finance:** Manage project costs by applying engineering management principles.

### **Programme Specific Outcome (PSOs):**

* **PSO1:** Apply knowledge of cost estimation models to develop sustainable IT solutions.

### **Result & Discussion:**

* **Result:** Project cost estimation completed using the selected technique/model.
* **Discussion:**
  + The selected cost estimation technique provided detailed cost breakdowns for various project components.
  + The estimate highlighted key cost-driving factors, enabling resource optimization.

### **Conclusion:**

Cost estimation is a vital step in software project planning, ensuring proper allocation of resources and project success. Through this experiment, students gained practical experience in applying cost estimation techniques and understanding the factors affecting project costs.

### **Steps for the Experiment**

#### ****1. Understand Project Requirements****

1. Identify the size and scope of the project.
2. Define the functional and non-functional requirements clearly.

#### ****2. Choose a Cost Estimation Technique****

1. Select a suitable cost estimation technique such as COCOMO, FPA, or Expert Judgment.
2. Justify the choice of technique based on project characteristics.

#### ****3. Perform Cost Estimation****

1. Calculate project size (e.g., LOC, function points).
2. Use estimation formulas or tools based on the chosen technique.
3. Break down costs into subcomponents like development, testing, and deployment.

#### ****4. Document and Evaluate Results****

1. Create a detailed cost estimation report.
2. Verify the estimates against project requirements and constraints.
3. Suggest ways to optimize costs, if applicable.

### **Tools for Cost Estimation**

* COCOMO Calculator
* Function Point Analysis Tools
* Estimation spreadsheets or software (e.g., MS Excel)