Department of Computer

Academic Term: First Term 2023**Class: T.E /Computer Sem – V /**

Software Engineering

Practical No:	**4**
Title:	**Calculating Function Points of the Project in Software Engineering**
Date of Performance:	**06-08-23**
Roll No:	**9639**
Team Members:	**Jenny Lopes and Janvi Naik**

Rubrics for Evaluation:

Sr. No	Performance Indicator	Excellent	Good	Below Average	Total Score
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Theory Understanding(02)	02(Correct)	NA	01 (Tried)	
3	Content Quality (03)	03(All used)	02 (Partial)	01 (rarely followed)	

4	Post Lab Questions (04)	04(done well)	3 (Partially Correct)	2(submitted)	

Signature of the Teacher:

Lab Experiment 04

Experiment Name: Calculating Function Points of the Project in Software Engineering

Objective: The objective of this lab experiment is to introduce students to the concept of Function Points and the Function Point Analysis (FPA) technique for measuring software size and complexity. Students will gain practical experience in calculating Function Points for a sample software project, enabling them to estimate development effort and assess project scope accurately.

Introduction: Function Points are a unit of measurement used in software engineering to quantify the functionality delivered by a software application. Function Point Analysis (FPA) is a widely used technique to assess the functional size of a software project based on its user requirements.

Lab Experiment Overview:

1. Introduction to Function Points: The lab session begins with an overview of Function Points, explaining the concept and their significance in software size measurement.
2. Defining the Sample Project: Students are provided with a sample software project along with its user requirements. The project may involve modules, functionalities, and user interactions.
3. Identifying Functionalities: Students identify and categorize the functionalities in the sample project, such as data inputs, data outputs, inquiries, external interfaces, and internal logical files. 4. Assigning Complexity Weights: For each identified functionality, students assign complexity weights based on specific criteria provided in the FPA guidelines.
5. Calculating Unadjusted Function Points: Students calculate the Unadjusted Function Points (UFP) by summing up the weighted functionalities.
6. Adjusting Function Points: Students apply adjustment factors (e.g., complexity, performance, and conversion) to the UFP to calculate the Adjusted Function Points (AFP).
7. Estimating Development Effort: Using historical data or industry benchmarks, students estimate the development effort required for the project based on the calculated AFP. 8. Conclusion and Reflection: Students discuss the significance of Function Points in software estimation and reflect on their experience in calculating Function Points for the sample project.

Learning Outcomes: By the end of this lab experiment, students are expected to:

· Understand the concept of Function Points and their role in software size measurement. · Gain practical experience in applying the Function Point Analysis (FPA) technique to assess software functionality.
· Learn to categorize functionalities and assign complexity weights in Function Point calculations. · Develop estimation skills to assess development effort based on calculated Function Points. ·

Appreciate the importance of accurate software size measurement in project planning and resource allocation.

Pre-Lab Preparations: Before the lab session, students should familiarize themselves with the concept of Function Points and the guidelines for their calculation. They should review the

categorization of functionalities and the complexity weighting factors used in Function Point Analysis.

Materials and Resources:

· Project brief and details for the sample software project
· Function Point Analysis guidelines and complexity weighting criteria ·
Calculators or spreadsheet software for performing calculations

Creating a mental health support chatbot is a meaningful project that can have a positive impact on people's well-being. Below, I've outlined a project brief and details for this sample software project, provided Function Point Analysis guidelines and complexity weighting criteria, and mentioned the need for calculators or spreadsheet software for performing calculations related to this project.

1. Project Brief and Details: Mental Health Support Chatbot

Developing a chatbot software application designed to provide mental health support to users. The chatbot will offer emotional support, provide resources, and guide users through various exercises to improve their mental well-being. The primary goal is to create a user-friendly and empathetic virtual companion that can help users cope with stress, anxiety, and other mental health issues.

1. Creating a chatbot that can engage in natural language conversations with users.

2. Providing emotional support, empathy, and active listening.

3. Offering self-help resources, such as articles, videos, and guided exercises.

4. Implementing data privacy and security measures to protect user information.

5. Monitoring and analyze user interactions to improve the chatbot's responses over time.

6. Ensuring accessibility for users with disabilities.

7. Deploying the chatbot on multiple platforms (web, mobile apps, messaging apps).

2. Function Point Analysis (FPA) Guidelines:

Function Point Analysis is a method to measure the functionality of software applications based on user interactions. For this project, you can use FPA to estimate the size and complexity of the chatbot system. Here are some FPA guidelines:

1. External Inputs (EI):

- Each type of user input or request (e.g., asking for advice, sharing feelings) should be counted.

- Complexity can be low for simple requests and high for complex therapy sessions.

- User initiates a conversation (Average Complexity) - 4

- User asks for advice (Average Complexity) - 4

- User shares feelings (High Complexity) - 6

2. External Outputs (EO)

- Each significant piece of information the chatbot provides as output (e.g., advice, resources) should be counted.

- Complexity can vary based on the depth and relevance of the information.

- Chatbot provides emotional support (Average Complexity) - 4

- Chatbot shares self-help articles (Average Complexity) - 4

- Chatbot offers relaxation exercises (High Complexity) - 6

3. External Inquiries (EQ):

- Each inquiry or query made to external databases or resources (e.g., retrieving articles) should be counted.

- Complexity depends on the complexity of the query and response.

- Chatbot retrieves mental health resources (Average Complexity) - 4 - Chatbot queries user preferences (Low Complexity) - 3

4. Internal Logical Files (ILF):

- Count data maintained by the chatbot (e.g., user profiles, conversation history).
- Complexity depends on the number of data elements and their relationships. - User profiles and history (Average Complexity) - 4

5. External Interface Files (EIF):

- Count any external data files that the chatbot interacts with (e.g., a database of mental health resources).
- Complexity depends on the number of files and data elements. - Database of mental health resources (Low Complexity) - 3

Complexity Weighting Criteria:

Assign complexity weights to each of the Function Point Analysis components based on the following scale:

- Low Complexity (3 points): Simple interactions, minimal data processing.
- Average Complexity (4 points): Moderately complex interactions, moderate data processing.
- High Complexity (6 points): Complex interactions, extensive data processing, integration with external systems.

3. Now, calculate the Unadjusted Function Points (UFP) for each category by multiplying the counts by their complexity weights and summing them up:

UFP (External Inputs) = 4 (User initiates) + 4 (User asks for advice) + 6 (User shares feelings) = 14

UFP (External Outputs) = 4 (Emotional support) + 4 (Self-help articles) + 6 (Relaxation exercises)= 14

UFP (External Inquiries) = 4 (Retrieve resources) + 3 (Query user preferences) = 7

UFP (Internal Logical Files) = 4 (User profiles and history) = 4

UFP (External Interface Files) = 3 (Database of resources) = 3

Next, sum up all the UFP values to get the Total Unadjusted Function Points (TUFP):

TUFP = 14 (EI) + 14 (EO) + 7 (EQ) + 4 (ILF) + 3 (EIF) = 42

Now, you need to adjust the UFP based on complexity factors like communication, processing logic, and data management. Apply an adjustment factor, which is typically determined based on organizational or project-specific factors. For this example, let's assume the adjustment factor is 1.2.

AFP (Adjusted Function Points) = TUFP * Adjustment Factor = 42 * 1.2 = 50.4

Postlab: Calculating function points of the Project:

a) Critically evaluate the Function Point Analysis method as a technique for software sizing and estimation, discussing its strengths and weaknesses.

Strengths of Function Point Analysis:

Objective Sizing: FPA provides an objective and consistent way to measure the functional size of software, regardless of the technology or development approach used. It focuses on the functionality delivered to the user, making it technology-independent.

User-Centric: FPA emphasizes the viewpoint of the end-users, focusing on the functions and features that directly contribute to their needs. This perspective aligns well with user satisfaction and business value.

Weaknesses of Function Point Analysis:

Complexity: FPA can be complex to implement, particularly for large and complex software systems. The process of counting function points involves several rules and can be time-consuming.

Subjectivity: Although FPA aims to be objective, some level of subjectivity can still be introduced, especially when categorizing transactions and assigning complexity values.

Learning Curve: FPA requires specialized training and expertise. Team members need to invest time and effort to become proficient in the method.

b) Apply the Function Point Analysis technique to a given software project and determine the function points based on complexity and functionalities.

Function Point Analysis (FPA) is a method for measuring the functional size of a software project based on the complexity and functionalities it offers. To apply FPA to a software project, you need to follow a structured process that involves identifying and classifying different types of functions within the software. Here's a simplified example of how you might apply FPA to estimate function points for a project:

Assume you're working on a simple e-commerce website. You need to consider two primary types of functions: External Inputs (EIs) and External Outputs (EOs). These functions are characterized by the number of data elements processed or referenced and the complexity of the processing.

External Inputs (EIs): These functions represent the user's interaction with the system to provide data. In an e-commerce website, this could be a customer placing an order.

For an EI, you need to consider:

- The number of data elements involved in the process.
- The complexity of the processing (low, average, or high).

Let's say that when a customer places an order, they enter information such as name, address, payment details, and a list of products. This process involves several data elements, and you consider it to have average complexity.

External Outputs (EOs): These functions represent the data the system sends to the user. In an e-commerce website, this could be generating an order confirmation for the customer.

For an EO, you need to consider:

- The number of data elements sent to the user.
- The complexity of the processing (low, average, or high).

Let's say that the order confirmation includes the customer's name, shipping details, and a list of products. This process involves several data elements, and you consider it to have low complexity.

Next, you would determine the total function points by adding up the values for each function. In this simplified example:

- Total EIs = 1 (customer order)
- Total EOs = 1 (order confirmation)

Now, consider the complexity adjustment. The complexity of EIs and EOs was rated as average and low, respectively. Use the weights specified by the IFPUG (International Function Point Users Group) or other relevant standards to assign complexity values. For example:

- Complexity Weight for Average EI = 4
- Complexity Weight for Low EO = 5

Now, calculate the unadjusted function points:

- Unadjusted Function Points = (Total EIs × Complexity Weight for EI) + (Total EOs × Complexity Weight for EO)
- Unadjusted Function Points = (1 × 4) + (1 × 5)
- Unadjusted Function Points = 4 + 5
- Unadjusted Function Points = 9

These unadjusted function points represent the size of your software project based on its functionalities and complexity. Keep in mind that this is a highly simplified example. In a real-world scenario, you would have more function types (e.g., External Inquiries, Internal Logical Files) and more functions to account for. You would also use the actual complexity weights from the IFPUG guidelines or other relevant standards for a more accurate estimate.

c) Propose strategies to manage and mitigate uncertainties in function point estimation and how they can impact project planning and resource allocation.

Managing and mitigating uncertainties in function point estimation is essential for effective project planning and resource allocation. Here are several strategies to address uncertainties in function point estimation and understand their impact on project planning and resource allocation:

Use a Range of Estimates: Rather than providing a single point estimate, consider presenting a range of estimates. This range can indicate the minimum, most likely, and maximum estimates. This approach acknowledges the inherent uncertainty in software development and provides decision-makers with more information.

Historical Data Analysis: Review historical data from past projects that used function point estimation. This can help in identifying trends and patterns related to estimation accuracy. Use this data to refine your estimates based on past performance.

Sensitivity Analysis: Conduct sensitivity analysis to identify which variables or assumptions have the most significant impact on your estimates. By understanding the sensitivities, you can focus on managing those aspects more effectively.

Impact on Project Planning and Resource Allocation:

Accurate Planning: Mitigating uncertainties in function point estimation leads to more accurate project planning. When you have a better understanding of potential variations, you can set realistic goals and deadlines.

Resource Allocation: By addressing uncertainties, you can allocate resources more effectively. Contingency planning allows you to allocate additional resources or time for potential setbacks, reducing the risk of resource shortages.

Risk Management: Identifying and managing uncertainties helps with risk management. When you anticipate potential issues, you can proactively plan for them, which can reduce the impact of unexpected problems on project timelines and budgets.

Conclusion: The lab experiment on calculating Function Points for a software project provides students with a practical approach to estimating software size and complexity. By applying the Function Point Analysis (FPA) technique, students gain insights into the importance of objective software measurement for project planning and resource allocation. The hands-on experience in identifying functionalities and assigning complexity weights enhances their estimation skills and equips them with valuable techniques for effective project management. The lab experiment encourages students to apply Function Point Analysis in real-world scenarios, promoting accuracy and efficiency in software size measurement for successful software engineering projects.