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:	**06-09-23**
Roll No:	**9618**
Team Members:	**Jenny Lopes, Janvi Naik, Anuf Shaikh**

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

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.

POSTLABS:

Q1) Critically evaluate the Function Point Analysis method as a technique for software sizing and estimation, discussing its strengths and weaknesses.

Function Point Analysis (FPA) is a widely used technique for software sizing and estimation. It is a structured method that quantifies the functionality provided by a software application and has been in use for several decades. Here, I'll provide a critical evaluation of FPA by discussing its strengths and weaknesses.

Strengths:

Independence from Implementation Details: FPA is focused on measuring the functionality of a software system, making it independent of the underlying technology or programming language used. This allows for consistent measurement across different technologies and platforms.

User-Centric: FPA measures the functionality in terms of what it delivers to the end-users. This user-centric approach helps in understanding the software's value from a user's perspective, which can be vital in requirement analysis.

Accurate Sizing: When applied correctly, FPA provides a relatively accurate size estimate for a software project. This estimation is based on an understanding of the system's user requirements and can be used to estimate project effort and resources.

Historical Data: Over time, organizations can build a historical database of FPA measurements for various projects. This data can be invaluable for future estimation, benchmarking, and process improvement.

Facilitates Communication: FPA provides a common language for business analysts, developers, and project managers to discuss and understand the functional requirements and scope of a project. This can improve communication and reduce misunderstandings.

Supports Project Control: FPA can help in project control by providing a baseline for monitoring and controlling changes in project scope. It can also be used to track project progress and compare it with the initial estimates.

Weaknesses:

Complexity: FPA can be complex and time-consuming. It involves a multi-step process that requires a trained and experienced analyst to accurately determine the function points. The learning curve can be steep, and mistakes in counting can lead to inaccurate estimates.

Subjectivity: The process of counting function points involves a degree of subjectivity. Different analysts might interpret the same requirements differently, leading to variations in estimates.

Inaccuracy for Small Projects: FPA is more accurate for larger projects with a well-defined scope. For smaller projects or those with rapidly changing requirements, the effort required to perform FPA might not be justified, and the results may not be as accurate.

Limited Scope: FPA primarily focuses on the functional aspects of a software system. It does not account for non-functional attributes such as performance, security, and usability. These aspects are critical for many software projects but are not covered by FPA.

Not Agile-Friendly: FPA is less suitable for agile development methodologies, which emphasize flexibility and frequent changes in requirements. FPA's rigidity and focus on up-front planning can be a poor fit for agile projects.

Dependency on Accurate Requirements: FPA heavily relies on having a complete and accurate set of requirements, which can be a challenge in many software projects, especially early in the development process.

In conclusion, Function Point Analysis is a valuable technique for software sizing and estimation, especially for larger projects with well-defined requirements and a historical data repository. However, it is not without its challenges, including complexity, subjectivity, and limited coverage of non-functional aspects. Organizations should carefully consider their project's characteristics and requirements before deciding to use FPA as a primary estimation method.

Q2) 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 structured method for determining the size of a software project based on its functionality and complexity. To calculate function points, you need to consider various aspects of the software, such as user inputs, outputs, inquiries, files, and interfaces. The final result is a measure of the project's size in terms of function points. Here, I'll provide a simplified example to demonstrate how to calculate function points for a hypothetical software project.

Let's assume we have a simple project for a library management system. To calculate function points, you'll follow these steps:

Step 1: Identify Components

Identify the main components of the software system, which include:

External Inputs (EI): These are user-initiated actions that provide data to the system. For example, adding a new book to the library system.

External Outputs (EO): These are user-initiated actions that result in data being sent out from the system. For example, generating a list of overdue books.

External Inquiries (EQ): These are user-initiated actions that both input and output data. For example, searching for a book in the system.

Internal Logical Files (ILF): These are internal data files that the system maintains. For example, the database of books and their details.

External Interface Files (EIF): These are files shared with other systems. For example, an interface with a university database to verify student records.

Step 2: Assign Complexity Weights

Assign complexity weights to each of these components based on their complexity. FPA typically uses a scale of Low, Average, and High complexity. The complexity is determined by the number of data elements and their complexity.

For our example:

External Inputs (EI): Low complexity
External Outputs (EO): Average complexity
External Inquiries (EQ): High complexity
Internal Logical Files (ILF): Average complexity
External Interface Files (EIF): Low complexity
Step 3: Count the Components

Count the number of each component within the system. For our example:

EI: 5
EO: 3
EQ: 2
ILF: 3
EIF: 1
Step 4: Calculate Function Points

Now, use the following formula to calculate the unadjusted function points (UFP):

UFP = (EI * Weight_EI) + (EO * Weight_EO) + (EQ * Weight_EQ) + (ILF * Weight_ILF) + (EIF * Weight_EIF)

Substitute the values from your project:

UFP = (5 * 3) + (3 * 4) + (2 * 6) + (3 * 7) + (1 * 5) = 15 + 12 + 12 + 21 + 5 = 65 UFP

Step 5: Adjust for General System Characteristics (GSCs)

Depending on factors like data communication, distributed data processing, performance, and reusability, you can adjust the UFP. Each factor has a weight, and you may add or subtract points based on their influence. This step is more detailed and requires additional information and analysis.

Step 6: Calculate Adjusted Function Points

Adjusted Function Points (AFP) = UFP + Total GSC points

This will give you the final count of function points for your project.

Please note that this is a simplified example, and real-world projects may involve a more detailed analysis, especially in the adjustment for GSCs. Additionally, using FPA software or tools can help automate the process and reduce the margin for error.

Q3) 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 critical for accurate project planning and resource allocation. Function Point Analysis (FPA) can be subject to various sources of uncertainty, including ambiguous requirements, subjective judgment, and changes during the project. Here are some strategies to manage and mitigate these uncertainties and their potential impact on project planning and resource allocation:

1. Clearly Defined Requirements:

Uncertainty often arises from vague or changing requirements. Ensure that the project requirements are well-documented, clear, and unambiguous.
Involve domain experts, business analysts, and end-users to create a comprehensive and well-understood requirements document.
2. Early and Continuous Involvement:

Involve key stakeholders early in the project to get their input on functional requirements and validate assumptions.
Regularly communicate with stakeholders to address changing requirements and evolving project needs.
3. Use Historical Data:

Leverage historical data from previous projects to inform your function point estimation. Past projects can provide insights into the level of uncertainty encountered and how it impacted resource allocation.
4. Sensitivity Analysis:

Conduct sensitivity analysis by assessing the potential impact of varying assumptions and uncertain factors on the estimation.

Create scenarios with different sets of assumptions and evaluate their consequences on project planning and resource allocation.

5. Benchmarking:

Compare your function point estimates with industry benchmarks or similar projects to identify outliers or discrepancies that may indicate potential issues.

6. Risk Management:

Develop a comprehensive risk management plan to address potential uncertainties that may affect function point estimation. This plan should include risk identification, assessment, and mitigation strategies.

7. Review and Validation:

Conduct reviews and validations of your function point estimation with multiple stakeholders, including technical experts, project managers, and business analysts.
Encourage independent reviews to reduce bias and confirm the accuracy of the estimation.

8. Agile Methodologies:

Consider using Agile development methodologies that allow for flexibility and adaptation to changing requirements. Agile practices can help mitigate the impact of uncertainty on project planning.

9. Buffer Allocation:

Allocate contingency buffers in project planning to account for uncertainty in function point estimation. These buffers can cover both time and resources.

10. Communication and Documentation:

Clearly document all assumptions, decisions, and uncertainties in the estimation process. This documentation provides a basis for later analysis and learning from previous projects.
Impact on Project Planning and Resource Allocation:

Managing and mitigating uncertainties in function point estimation has a direct impact on project planning and resource allocation:

Accurate Scheduling: By reducing uncertainties, you can create more accurate project schedules, leading to better time management and project control.

Resource Allocation: With reduced uncertainty, you can allocate resources more efficiently. A more accurate estimation helps in determining the appropriate team size and skill set required for the project.

Cost Control: Effective uncertainty management helps control project costs by avoiding unexpected scope changes and resource shortages.

Risk Mitigation: By addressing uncertainties proactively, you reduce the risk of project delays, cost overruns, and quality issues.

Client Satisfaction: Accurate planning and resource allocation lead to meeting client expectations and delivering the project on time and within budget.

In summary, effective management of uncertainties in function point estimation is essential for successful project planning and resource allocation. It reduces the risk of project failure, improves stakeholder satisfaction, and enhances project control.