Department of Computer Engineering

Academic Term: First Term 2023-24

Class: T.E /Computer Sem – V / Software Engineering

Practical No:	4
Title:	Calculating function points of the Project
Date of Performance:	14/08/23
Roll No:	9568
Team Members:	Glison Pereira(9568), Pradyumnaa Kadam(9547), Shubham Ghadigaonkar(9606)

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.

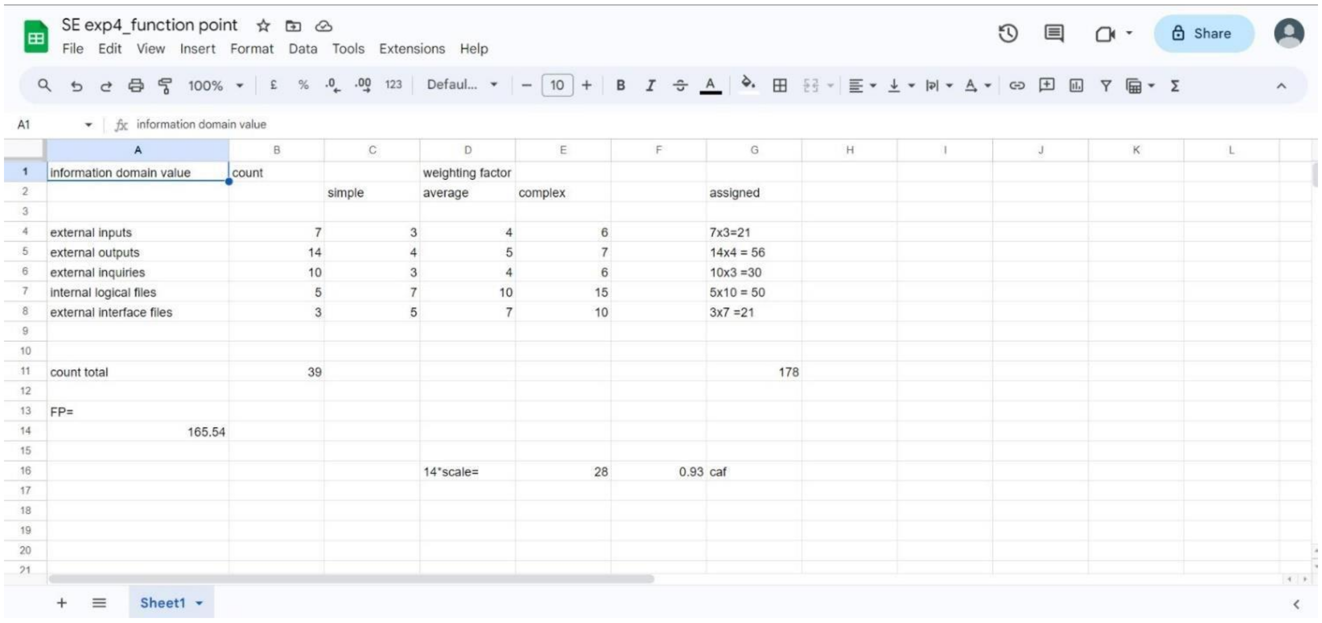
Pre-Lab Preparations: Before the lab session, students should familiarize themselves with the concept of Function Points and the guidelines for their calculation. They should review the

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

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.


SE exp4_function point
File Edit View Insert Format Data Tools Extensions Help
Share
A1 information domain value
Sheet1

	A	B	C	D	E	F	G
1	information domain value	count		weighting factor			
2			simple	average	complex		assigned
3							
4	external inputs	7	3	4	6		7x3=21
5	external outputs	14	4	5	7		14x4 = 56
6	external inquiries	10	3	4	6		10x3 =30
7	internal logical files	5	7	10	15		5x10 = 50
8	external interface files	3	5	7	10		3x7 =21
9							
10							
11	count total	39					178
12							
13	FP=						
14	165.54						
15							
16				14*scale=	28	0.93	caf

Postlab:

a) 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 technique used for software sizing and estimation. Strengths of Function Point Analysis (FPA):

1. Technology Agnostic: FPA is technology-agnostic, meaning it doesn't depend on the programming language, development tools, or platforms used. This makes it applicable to a wide range of software projects.

2. Focus on User's Perspective: FPA assesses software from the user's perspective by quantifying the functionality that the software delivers. This helps in aligning software development with user needs and expectations.

3. Considers Different Aspects: FPA takes into account various aspects of software functionality, including inputs, outputs, inquiries, internal logical files, and external interface files. This comprehensive approach ensures a holistic view of software complexity.

4. Simplicity and Understandability: The FPA method uses relatively simple and intuitive counting rules, making it accessible to a wide range of stakeholders, including non-technical ones. This facilitates communication between developers, project managers, and clients.

5. Historical Data: Over time, organizations can accumulate historical data on FPA counts and actual development effort, enabling better estimation accuracy through the use of historical productivity metrics.
6. Independent Verification: FPA can serve as an independent and objective measure of software size, which can be useful in contract negotiations and assessing vendor deliverables.

Weaknesses of Function Point Analysis (FPA):

1. Subjectivity: Despite its structured rules, FPA can be somewhat subjective. Different analysts may interpret and assess software functionality differently, leading to variations in size estimates.

2. Complexity in Measurement: FPA can become complex and time-consuming for large and complex software systems. Counting all the function points accurately may require substantial effort and expertise.

3. Initial Learning Curve: Training and experience are required to become proficient in FPA, and not all organizations invest in this training. This can lead to inaccurate estimates when performed by inexperienced analysts.

4. Limited to Functional Size: FPA primarily measures functional size and does not consider other factors that can impact development effort, such as technical complexity, architectural considerations, or the skill level of the development team.

5. Lack of Precision: FPA provides a rough estimate of software size but doesn't account for variations in development productivity or specific project risks. It's a sizing technique and not a project management or risk assessment method.

6. Not Suitable for All Projects: FPA may not be suitable for projects with a heavy emphasis on non-functional requirements (e.g., performance, security, scalability), as it focuses primarily on functional aspects.

7. Overemphasis on Documentation: Critics argue that FPA may incentivize excessive documentation in software development, as more documentation could lead to a higher function point count.

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 detailed process that involves assessing a software application's functionality based on various criteria. To determine function points, you'll need specific information about the software project, such as its requirements and features. Below is a simplified example of how FPA might be applied to a hypothetical software project.

Step 1: Identify User Inputs (UI):

- User Registration Form
- Login Form
- Search for Products
- Add Product to Cart
- Remove Product from Cart
- Update User Profile
- View Order History
- Place an Order
- Contact Customer Support

Step 2: Identify User Outputs (UO):

- Registration Confirmation
- Login Confirmation
- Search Results
- Shopping Cart Contents
- Order Confirmation
- User Profile Information
- Order History
- Order Receipt
- Customer Support Response

Step 3: Identify User Inquiries (UI):

- Forgot Password
- Product Information Request
- Shipping Information Request
- Return/Refund Request - Order Status Inquiry

Step 4: Identify Internal Logical Files (ILF):

- User Database
- Product Database
- Order Database
- Shopping Cart Data

Step 5: Identify External Interface Files (EIF):

- Payment Gateway Integration
- Shipping Carrier Integration - Product

Information API Step 6: Assess Complexity:

For each identified component (UI, UO, UI, ILF, EIF), assess their complexity based on the following scale: Low, Average, or High.

For example:

- Login Form (UI): Low Complexity
- Order Confirmation (UO): Average Complexity
- Product Database (ILF): High Complexity
- Shipping Carrier Integration (EIF): Average

Complexity Step

7: Calculate Function Points:

1. Count the total number of components in each category:
   - Total UI = 9
   - Total UO = 9
   - Total UI = 5
   - Total ILF = 4 - Total EIF = 3

2. Assign a weight to each component based on complexity (Low = 3, Average = 4, High = 6):
   - Weighted UI = (9 * 3) = 27
   - Weighted UO = (9 * 4) = 36
   - Weighted UI = (5 * 3) = 15
   - Weighted ILF = (4 * 6) = 24 - Weighted EIF = (3 * 4) = 12

3. Calculate the unadjusted function point (UFP) by summing the weighted components:
   - UFP = 27 + 36 + 15 + 24 + 12 = 114

Step 8: Account for Complexity Adjustment:

Determine the complexity adjustment factor (CAF) based on the overall complexity of the project. This factor typically ranges from 0.65 to 1.35, with 1.0 being average complexity.

Step 9: Calculate Adjusted Function Points: Adjusted

Function Points (AFP) = UFP * CAF

For example, if CAF = 1.2:
AFP = 114 * 1.2 = 136.8 (round to the nearest whole number, so AFP = 137)

Step 10: Use Function Points for Estimation:

You can use the calculated function points to estimate various aspects of the project, such as development effort, project duration, and resource allocation, by applying industry-specific productivity metrics and historical data.

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

Function Point Estimation (FPE) is a valuable technique for estimating software size and, by extension, project effort and resource allocation. However, like any estimation method, it is subject to uncertainties. Here are strategies to manage and mitigate these uncertainties and how they can impact project planning and resource allocation:

1. Expert Involvement:

- Strategy: Involve experienced FPA practitioners in the estimation process.
- Impact: Experts can provide more accurate assessments of function points, identify potential complexities, and ensure that the estimation process follows best practices.

2. Detailed Requirements Analysis:

- Strategy: Ensure thorough and detailed requirements analysis before estimating function points.
- Impact: Clear and well-defined requirements reduce uncertainty in FPE. Incomplete or ambiguous requirements can lead to inaccurate estimations.

3. Historical Data:

- Strategy: Use historical data from similar past projects as a reference.
- Impact: Historical data provides benchmarks for estimation accuracy, helping in better resource allocation and project planning.

6. Range Estimations:

- Strategy: Provide a range of function point estimates (e.g., optimistic, pessimistic, most likely).
- Impact: Ranges acknowledge uncertainty and allow for more flexible project planning and resource allocation. Contingency plans can be developed based on different scenarios.

7. Sensitivity Analysis:

- Strategy: Conduct sensitivity analysis to identify how changes in function point estimates affect project outcomes.
- Impact: Sensitivity analysis helps in understanding the impact of estimation uncertainties on project planning, allowing for better risk management.

8. Risk Mitigation Plans:

- Strategy: Develop risk mitigation plans for potential estimation uncertainties.
- Impact: Having contingency plans in place ensures that the project can adapt to unexpected changes without major disruptions in resource allocation.

9. Regular Reviews and Updates:

- Strategy: Periodically review and update function point estimates as the project progresses and more information becomes available.
- Impact: Ongoing estimation adjustments can lead to more accurate resource allocation and better project planning, especially in agile environments.

10. Communication and Transparency:

- Strategy: Communicate the level of uncertainty associated with function point estimates to stakeholders.
- Impact: Transparent communication ensures that stakeholders understand the limitations of estimations and can make informed decisions about resource allocation and project planning.