**Chapter 3**

<https://www.slideshare.net/arafkarsh/function-point-analysis-65711721>

COCOMO

<https://www.geeksforgeeks.org/software-engineering-application-composition-estimation-model-cocomo-ii-stage-1/>

<https://slideplayer.com/slide/14811225/>

EVA

<https://www.geeksforgeeks.org/what-is-earned-value-analysis-eva/>

<https://youtu.be/7qGUyTfjWCw?si=nJg7KMbRVaCxjzA4>

<https://www.slideshare.net/tranquangminhtan/software-estimation-techniques-23263528>

**Cost estimation**

**how to perform Function Point Analysis and related calculations.**

**here are the formulas for each of the concepts along with examples:**

1. \*\*Unadjusted Function Points (UFP):\*\*

Formula: UFP = Σ(Count of each type of transaction \* Weight for that type)

Example: UFP = (22 x 4) + (45 x 5) + (6 x 7) + (5 x 10) + (2 x 7) = 419

2. \*\*Processing Factors in Adjusted Function Points (AFP):\*\*

Formula: AFP = UFP x [0.65 + (0.01 x Σ(Processing Factors))]

Example: AFP = 419 x [0.65 + (0.01 x (5 + 1 + 0 + 4 + 3 + 5 + 4 + 3 + 4 + 5 + 2 + 3 + 4 + 2))] = 461

3. \*\*Productivity:\*\*

Formula: Productivity = AFP / Effort in Person-Months (P/M)

Example: Productivity = 461 / 37 P/M ≈ 12.46 Function Points per P/M

4. \*\*Documentation per Function Point:\*\*

Formula: Documentation per Function Point = Total Documentation / AFP

Example: If total documentation is 370 pages, Documentation per Function Point = 370 / 461 ≈ 0.80 pages per function point

5. \*\*Cost per Function Point:\*\*

Formula: Cost per Function Point = Total Cost / AFP

Example: If the total cost is $9,520, Cost per Function Point = $9,520 / 461 ≈ $20.64 per function point

**Questions ?**

1. **Define Unadjusted Function Points (UFP) and explain how to calculate them.**

Unadjusted Function Points (UFP) are a measure used in software development to quantify the size and complexity of a software application or project before any adjustments or modifications are made for factors such as technology, complexity, or environmental considerations. UFP serve as a foundation for estimating effort, cost, and resources required for software development and maintenance.

To calculate Unadjusted Function Points (UFP), you typically follow a structured process that involves identifying and quantifying five different types of functional user interactions or functionalities within the software:

1. External Inputs (EI): These are user interactions that result in data being entered or updated within the system. For each unique input, count it as one EI.

2. External Outputs (EO): These represent user interactions that result in data being displayed or sent out of the system. Count each unique output as one EO.

3. External Inquiries (EQ): EQs are user interactions that involve queries or requests for information from the system. Count each unique inquiry as one EQ.

4. Internal Logical Files (ILF): These are data stores or collections of related data within the system. Count each unique logical file as one ILF.

5. External Interface Files (EIF): EIFs are data stores or files that are used by the system but maintained by external applications or systems. Count each unique interface file as one EIF.

Once you have identified and counted these five functional types, assign a complexity rating to each of them based on a scale of low, average, or high complexity. The complexity rating depends on factors such as the number of data elements involved, the intricacy of processing, and the degree of user interaction.

After determining the complexity ratings, you can calculate the Unadjusted Function Points (UFP) using the following formula:

UFP = (EI \* C\_EI) + (EO \* C\_EO) + (EQ \* C\_EQ) + (ILF \* C\_ILF) + (EIF \* C\_EIF)

Where:

- UFP is the Unadjusted Function Points.

- EI, EO, EQ, ILF, and EIF represent the counts of the respective functional types.

- C\_EI, C\_EO, C\_EQ, C\_ILF, and C\_EIF are the complexity ratings assigned to each functional type (usually 1 for low, 2 for average, and 3 for high complexity).

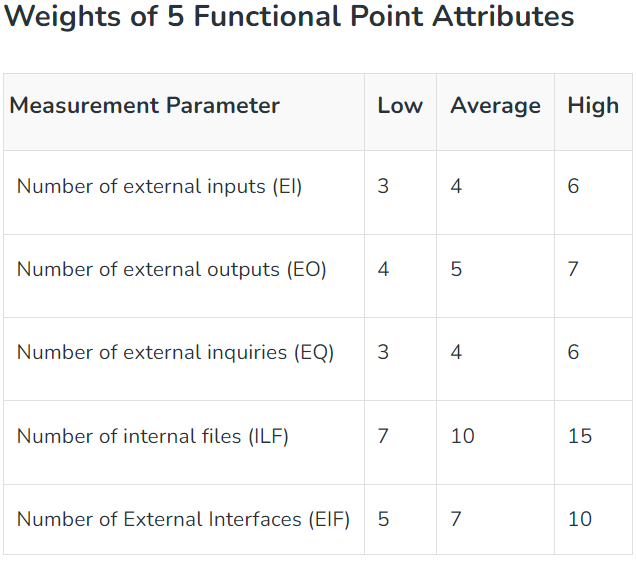
The result, UFP, provides a quantitative measure of the size and complexity of the software system based solely on its functional requirements. It serves as a valuable input for subsequent steps in the software development process, such as estimating effort, cost, and resource allocation.

1. **What are the different types of complexity in Function Point Analysis (FPA), and how are -they represented in the calculation?**

Function Point Analysis (FPA) is a method used in software engineering to measure the functional size of a software application. It quantifies the functionality provided by a software system, which helps in estimating the effort, cost, and resources required for development and maintenance. FPA considers several types of complexity when calculating function points. These complexities are represented in the calculation through different factors, which are multiplied by the count of various functional elements. The primary types of complexity in FPA are:

* External Input (EI) Complexity:
  + Simple: The external input has low complexity, and data is read from or written to a few data elements.
  + Average: The external input has moderate complexity, involving several data elements or some validation.
  + Complex: The external input is highly complex, involving many data elements and complex validation rules.
* External Output (EO) Complexity:
  + Simple: The external output involves presenting data to the user with low complexity.
  + Average: The external output involves some processing or formatting of data before presenting it to the user.
  + Complex: The external output is complex, requiring significant processing before presentation.
* External Inquiry (EQ) Complexity:
  + Simple: The external inquiry is straightforward and involves retrieving a small amount of data.
  + Average: The external inquiry involves more data retrieval and may have some complexity.
  + Complex: The external inquiry is complex, requiring substantial data retrieval and complex processing.
* Internal Logical File (ILF) Complexity:
  + Simple: The internal logical file is simple, containing a few data elements.
  + Average: The internal logical file is moderately complex, with several data elements.
  + Complex: The internal logical file is highly complex, containing many data elements and complex relationships.
* External Interface File (EIF) Complexity:
  + Simple: The external interface file is simple, with a small number of data elements.
  + Average: The external interface file is moderately complex, with several data elements.
  + Complex: The external interface file is highly complex, containing many data elements and complex relationships.

To calculate function points, you assign a weight to each type of complexity (Simple, Average, or Complex) for each functional element (EI, EO, EQ, ILF, EIF) in your software project. The weightings are predefined based on industry standards. You then count the instances of each functional element in your project and multiply them by the corresponding weight to calculate the Unadjusted Function Points (UFP). After considering factors like technical complexity and environmental factors, you arrive at the Adjusted Function Points (AFP). These adjusted function points serve as a measure of the size of the software application and can be used for estimating project effort and resources

.

For further reference

https://www.geeksforgeeks.org/software-engineering-functional-point-fp-analysis/

1. **Explain the concept of Processing Factors in Function Point Analysis. Provide examples of Processing Factors and their impact on the calculation.**

In Function Point Analysis (FPA), processing factors are used to adjust the Unadjusted Function Points (UFP) to account for various technical and environmental factors that can influence the complexity and effort required for software development. These factors help in refining the Function Point count and providing a more accurate estimate of the effort and resources needed for a project. Processing factors are also known as General System Characteristics (GSCs) and are applied to the UFP to calculate the Adjusted Function Points (AFP). Each processing factor is assigned a weight or rating that represents its impact on the project.

Here are some common examples of processing factors and their impact on the calculation:

**1. Data Communications (DC):** This factor accounts for the complexity of data exchange between the software being measured and external systems or components. The weight assigned to DC reflects the complexity of data input/output and communication protocols. For example:

- Low Complexity (DC = 1): Simple data exchanges, minimal data validation.

- Moderate Complexity (DC = 2): Moderate data validation and communication.

- High Complexity (DC = 3): Complex data exchanges, extensive validation, and multiple communication protocols.

**2. Distributed Data Processing (DDP):** DDP considers the distribution of data and processing across multiple locations or platforms. The weight for DDP depends on the complexity of data distribution and the integration of remote processing elements. For example:

- Low Complexity (DDP = 1): Minimal or no distributed processing, centralized data.

- Moderate Complexity (DDP = 2): Some distributed processing, moderate data sharing.

- High Complexity (DDP = 3): Extensive distributed processing, complex data sharing, and synchronization.

**3. Performance and Efficiency (PE):** This factor reflects the importance of performance and efficiency considerations in the software. It takes into account factors like response time requirements and resource utilization. For example:

- Low Complexity (PE = 1): No strict performance requirements, ample resources.

- Moderate Complexity (PE = 2): Moderate performance requirements, some resource constraints.

- High Complexity (PE = 3): Stringent performance requirements, resource optimization critical.

**4. Transaction Rate (TR):** TR deals with the volume and frequency of transactions processed by the software. Higher transaction rates may require additional complexity in design and implementation. For example:

- Low Complexity (TR = 1): Low transaction volume or frequency.

- Moderate Complexity (TR = 2): Moderate transaction volume or frequency.

- High Complexity (TR = 3): High transaction volume or frequency.

**5. Online Data Entry (ODE):** ODE reflects the complexity of online data entry and validation requirements. It considers factors like the number of data elements and the level of validation needed. For example:

- Low Complexity (ODE = 1): Simple online data entry, minimal validation.

- Moderate Complexity (ODE = 2): Moderate data entry complexity and validation.

- High Complexity (ODE = 3): Complex online data entry, extensive validation.

To calculate the Adjusted Function Points (AFP), you multiply the UFP by the weighted values of each processing factor and sum them up:

AFP = UFP \* (DC + DDP + PE + TR + ODE)

The AFP provides a more refined measure of the software's complexity, accounting for technical and environmental factors, and serves as a basis for estimating development effort, cost, and resource allocation.

1. **Calculate the Total Unadjusted Function Points (UFP) for a given set of data, including counts of External Inputs (EI), External Outputs (EO), External Inquiries (EQ), Internal Logical Files (ILF), and External Interface Files (EIF).**

To calculate the Total Unadjusted Function Points (UFP), you'll need to determine the count of each functional element type (External Inputs, External Outputs, External Inquiries, Internal Logical Files, and External Interface Files) and then assign the corresponding complexity weights (Simple, Average, or Complex) to each of them. After that, you can calculate the UFP by multiplying the counts by their respective complexity weights and summing them up.

Let's assume you have the following counts for each functional element:

External Inputs (EI): 10

External Outputs (EO): 8

External Inquiries (EQ): 5

Internal Logical Files (ILF): 6

External Interface Files (EIF): 4

Now, let's also assume that you have assessed the complexity for each of these functional elements as follows:

EI Complexity: 6 Simple, 3 Average, 1 Complex

EO Complexity: 4 Simple, 3 Average, 1 Complex

EQ Complexity: 3 Simple, 2 Average

ILF Complexity: 4 Simple, 2 Average

EIF Complexity: 3 Simple, 1 Average

Next, you'll multiply each count by its respective complexity weight and sum them up to calculate the UFP:

For External Inputs (EI):

6 (Simple) \* 10 (Count) = 60

3 (Average) \* 8 (Count) = 24

1 (Complex) \* 0 (Count) = 0

Total for EI = 60 + 24 + 0 = 84

For External Outputs (EO):

4 (Simple) \* 8 (Count) = 32

3 (Average) \* 8 (Count) = 24

1 (Complex) \* 0 (Count) = 0

Total for EO = 32 + 24 + 0 = 56

For External Inquiries (EQ):

3 (Simple) \* 5 (Count) = 15

2 (Average) \* 0 (Count) = 0

Total for EQ = 15 + 0 = 15

For Internal Logical Files (ILF):

4 (Simple) \* 6 (Count) = 24

2 (Average) \* 0 (Count) = 0

Total for ILF = 24 + 0 = 24

For External Interface Files (EIF):

3 (Simple) \* 4 (Count) = 12

1 (Average) \* 0 (Count) = 0

Total for EIF = 12 + 0 = 12

Now, sum up the totals for each functional element:

Total UFP = Total for EI + Total for EO + Total for EQ + Total for ILF + Total for EIF

Total UFP = 84 + 56 + 15 + 24 + 12

Total UFP = 191

So, the Total Unadjusted Function Points (UFP) for this set of data is 191.

(**Note:- This is the assumption according to question the data set should have been given**)

1. **Define Adjusted Function Points (AFP) and describe the formula used to calculate them, including the role of Processing Factors.**

Adjusted Function Points (AFP) are a measure used in Function Point Analysis (FPA) to quantify the size and complexity of a software application or project after considering various technical and environmental factors. AFP is calculated by adjusting the Unadjusted Function Points (UFP) using Processing Factors (also known as General System Characteristics or GSCs) that account for the impact of factors like data communication, distributed processing, performance requirements, transaction rates, and online data entry complexity.

The formula for calculating Adjusted Function Points (AFP) involves applying the weights of the Processing Factors to the UFP:

AFP = UFP \* (GSC1 + GSC2 + ... + GSCn)

Where:

- AFP is the Adjusted Function Points.

- UFP is the Unadjusted Function Points.

- GSC1, GSC2, ..., GSCn are the weights associated with each Processing Factor (General System Characteristic).

The Processing Factors (GSCs) have specific values assigned based on the complexity or impact of each factor on the project. These weights typically range from 0 to 5, with 0 indicating no impact or very low complexity and 5 indicating high complexity or a significant impact. The exact values may vary depending on the specific Function Point Analysis method being used.

Here's a brief description of the Processing Factors (GSCs) used in the calculation of AFP:

**1. Data Communications (DC):** Reflects the complexity of data exchanges between the software system and external components or systems. The weight assigned to DC indicates the complexity of data input/output and communication protocols.

**2. Distributed Data Processing (DDP):** Accounts for the distribution of data and processing across multiple locations or platforms. The weight reflects the complexity of data distribution and integration of remote processing elements.

**3. Performance and Efficiency (PE):** Considers the importance of performance and efficiency requirements in the software. The weight indicates the significance of factors like response time and resource utilization.

**4. Transaction Rate (TR):** Deals with the volume and frequency of transactions processed by the software. The weight reflects the complexity associated with handling different transaction rates.

**5. Online Data Entry (ODE):** Reflects the complexity of online data entry and validation requirements. The weight is based on factors such as the number of data elements and the level of validation needed.

By applying these Processing Factors to the UFP, you get the AFP, which provides a more refined measure of the software's size and complexity, considering various technical and environmental factors. AFP serves as a valuable input for estimating development effort, cost, and resource allocation for the software project, helping project managers make more accurate planning and resource allocation decisions.

1. **Calculate the Adjusted Function Points (AFP) for a given set of data using Processing Factors.**

In Function Point Analysis (FPA), Processing Factors are used to adjust the Unadjusted Function Points (UFP) to account for various factors that can influence the complexity and effort required for a software project. These factors are applied to the UFP to calculate the Adjusted Function Points (AFP), which provide a more accurate estimation of the project's size and complexity. Processing Factors are typically expressed as percentages and can increase or decrease the UFP based on the project's characteristics.

Here are some examples of Processing Factors and their impact on the calculation:

1. **Data Communications (DC):** This factor accounts for the complexity of data exchanges between the application being measured and other applications or systems. It reflects the effort needed to handle data transfers, such as through APIs or web services.
   * If there are no complex data communications involved, the DC factor might be 0%, indicating no adjustment to UFP.
   * If there are extensive and complex data communications, the DC factor might be 10% or higher, increasing the UFP to reflect the added effort.
2. **Distributed Data Processing (DDP):** DDP considers the complexity arising from the distribution of data across multiple locations or platforms. Projects dealing with data synchronization or distributed databases can have higher DDP values.
   * If the project involves straightforward data processing within a single location, the DDP factor might be 0%.
   * If the project deals with complex data synchronization between multiple locations or platforms, the DDP factor might be 5% or more.
3. **Performance Requirements (PER):** This factor takes into account performance constraints and response time requirements that may add complexity to the software.
   * If there are no stringent performance requirements, the PER factor might be 0%.
   * If the project requires extremely high performance and low response times, the PER factor might be 5% or higher.
4. **Heavily Used Configuration (HUC):** The HUC factor considers the complexity associated with supporting multiple configurations or environments.
   * If the application has a single configuration and deployment environment, the HUC factor might be 0%.
   * If the application needs to support various configurations and environments, the HUC factor might be 5% or more.
5. **Transaction Rate (TR):** TR reflects the impact of a high transaction rate on the application's complexity. Applications that need to handle a large volume of transactions per unit of time might have a higher TR factor.
   * For applications with a standard transaction rate, the TR factor might be 0%.
   * For applications with a very high transaction rate requirement, the TR factor might be 5% or more.
6. **Explain the concept of productivity in the context of Function Point Analysis. How is it calculated, and what does it measure?**

Productivity in the context of Function Point Analysis (FPA) is a measure of how efficiently a development team or organization is able to produce software by considering the amount of functionality delivered (in terms of Function Points) in relation to the effort expended (typically in person-hours or person-days). It helps assess the efficiency and effectiveness of software development processes and can be a useful metric for tracking and improving the performance of software development teams.

Productivity is calculated using the following formula:

Productivity = Function Points (FP) / Effort (person-hours or person-days)

Where:

- Function Points (FP) represent the size and complexity of the software as measured by FPA.

- Effort is the total number of person-hours or person-days required to develop the software.

Productivity in FPA measures how much functional value or functionality is delivered for each unit of effort invested. It provides insight into whether a development team is working efficiently and whether there is a good balance between the scope of the software being developed and the resources allocated to the project.

Here's how productivity in FPA is interpreted:

**1. High Productivity:** A high productivity value indicates that the development team is efficiently producing software. They are delivering a significant amount of functionality for each unit of effort expended. This can be a positive sign of effective development processes, skilled developers, or the use of efficient development tools and practices.

**2. Low Productivity:** A low productivity value suggests that the development team is less efficient in producing software. They are delivering less functionality for the same amount of effort or are expending more effort to achieve a given level of functionality. This may be due to various factors, such as inefficient processes, inadequate skills, or complex project requirements.

**3. Steady or Improving Productivity:** Monitoring productivity over time can help assess the effectiveness of process improvements or changes in development practices. A steady or improving trend in productivity can indicate that efforts to optimize development processes are yielding positive results.

**4. Declining Productivity:** A declining productivity trend may signal issues in the development process, such as increased complexity, scope creep, or resource constraints. It can serve as an early warning sign for potential problems in the project.

It's important to note that while productivity is a valuable metric, it should be used in conjunction with other performance indicators, such as quality, customer satisfaction, and project timelines, to get a comprehensive view of software development performance.

Additionally, when calculating productivity, it's essential to use consistent and accurate measurements of Function Points and effort. FPA provides a standardized way to measure software size and complexity, making it a useful tool for assessing and improving productivity in software development projects.

1. **Calculate productivity given the Adjusted Function Points (AFP) and Effort in Person-Months (P/M).**

Productivity in the context of software development is typically measured as the number of Adjusted Function Points (AFP) delivered per Person-Month (P/M) of effort. To calculate productivity, you can use the following formula:



Where:

* AFP (Adjusted Function Points) is the total function points adjusted for various factors, as determined using Function Point Analysis.
* Effort in Person-Months (P/M) is the total amount of effort expended on the project, usually measured in person-months. One person-month represents the work done by one person in one month.

To calculate productivity, simply divide the AFP by the effort in person-months:



For example, if you have an AFP of 500 and the project required an effort of 25 person-months, the productivity would be:



So, the productivity in this example is 20 Adjusted Function Points per Person-Month. This metric can help assess how efficiently a software project team is delivering functionality relative to the effort expended. Higher productivity values are generally desirable, as they indicate that more functionality is being delivered for each unit of effort.

1. **Calculate the Documentation per Function Point for a given set of data.**

To calculate the Documentation per Function Point (DPFP), you'll need two pieces of information: the total amount of documentation or documentation effort (e.g., in person-hours or pages), and the total number of Function Points (FP) for the software project. The formula for DPFP is straightforward:

DPFP = Total Documentation Effort / Total Function Points

Let's say you have the following data for a software project:

- Total Documentation Effort: 500 person-hours

- Total Function Points: 200

You can use these values to calculate the Documentation per Function Point (DPFP):

DPFP = 500 person-hours / 200 Function Points

DPFP = 2.5 person-hours per Function Point

So, the Documentation per Function Point for this software project is 2.5 person-hours per Function Point. This means, on average, it took 2.5 person-hours of documentation effort to document each Function Point within the project. This metric can help you assess the level of documentation effort relative to the size of the software, which can be useful for tracking documentation efficiency and resource allocation in the development process.

1. **Calculate the Cost per Function Point for a given set of data.**

To calculate the Cost per Function Point (CPFP), you need to know the total cost of a software project and the total number of Adjusted Function Points (AFP) delivered in that project. The formula to calculate CPFP is as follows:



Where:

* Total Project Cost is the total cost incurred in developing and maintaining the software project.
* Total AFP (Adjusted Function Points) is the total function points adjusted for various factors, as determined using Function Point Analysis.

For example, if the total project cost is $500,000 and the total AFP delivered is 10,000 AFP, you can calculate the CPFP as follows:



So, the Cost per Function Point in this example is $50 per Adjusted Function Point. This metric helps assess the cost efficiency of delivering functionality in a software project. Lower CPFP values are generally more favorable, as they indicate that each function point is being delivered at a lower cost.