Software Effort Estimation
Lecture-5

By: Usama Musharaf

Contents of Lecture-5

Software Effort Estimation Technique

- Top Down
- Bottom Up
- Function Point Analysis

Software Effort Estimation

Software Effort Estimation

Software effort estimation involves predicting the amount of effort (usually in person-hours, person-days, or person-months) required to complete a software development task or project.

It is crucial for project planning, resource allocation, and cost estimation.

Software Effort Estimation Techniques

Software Effort Estimation Techniques

1. Expert Judgment

Relying on experienced project managers or software developers to estimate the effort based on their experience with similar projects.

2. Analogy-Based Estimation

Comparing the current project to similar, completed projects and using the historical effort data to make predictions.

Software Effort Estimation Techniques

3. Top-Down Estimation

Estimating the overall project effort at a high level, then breaking it down into smaller components.

It is based on the project’s objectives and scope rather than individual tasks.

Software Effort Estimation Techniques

4. Bottom-Up Estimation

Breaking down the project into smaller tasks or components and estimating the effort for each task individually.

The sum of all these estimates gives the total effort.

Software Effort Estimation Techniques

5. Function Point Analysis (FPA)

FPA measures the functionality provided by the software based on inputs, outputs, user interactions, and interfaces.

This functional size is then used to estimate effort.

Software Effort Estimation Techniques

6. Use Case Points

This method focuses on the number and complexity of use cases within the system, estimating effort based on use case points derived from system interactions.

Software Effort Estimation Techniques

7. COCOMO (Constructive Cost Model)

A parametric model that predicts the effort based on project size (in terms of lines of code or function points) and various factors like team experience, tools, and project complexity.

COCOMO I: Original model, with basic, intermediate, and detailed versions.

COCOMO II: Updated version, includes features to account for reuse, modern development methodologies, and more sophisticated cost drivers.

Software Effort Estimation Techniques

8. Machine Learning Approaches

Applying machine learning algorithms such as regression models, neural networks, or decision trees to predict effort based on historical data from previous projects.

Software Effort Estimation Techniques

9. Wideband Delphi

A structured group decision-making technique where a group of experts independently estimates the effort, then revises their estimates in a group discussion, iterating until consensus is reached.

Key Factors Influencing Effort Estimation

Project size: Larger projects require more effort.

Team experience: A more experienced team typically requires less effort.

Complexity: Highly complex systems need more time and resources.

Technology: New or unfamiliar technologies increase development time.

Requirements clarity: Well-defined requirements reduce uncertainty.

Bottom up Approach

A Procedural Code-Oriented Approach

1-Estimate the number and type of software modules in the final system.

2-Estimate the SLOC of each identified module.

- Program Structure Diagram

3-Estimate the work content, taking into account complexity and technical difficulty.

- Estimate the work content, taking into account complexity and technical difficulty
- Subjective Judgment

4-Calculate the work-days effort.

Program Structure Diagram

The ***XYZ annual maintenance*** contracts subsystem for which **Person A** is responsible will have a transaction which sets up details of new annual maintenance contract customers.

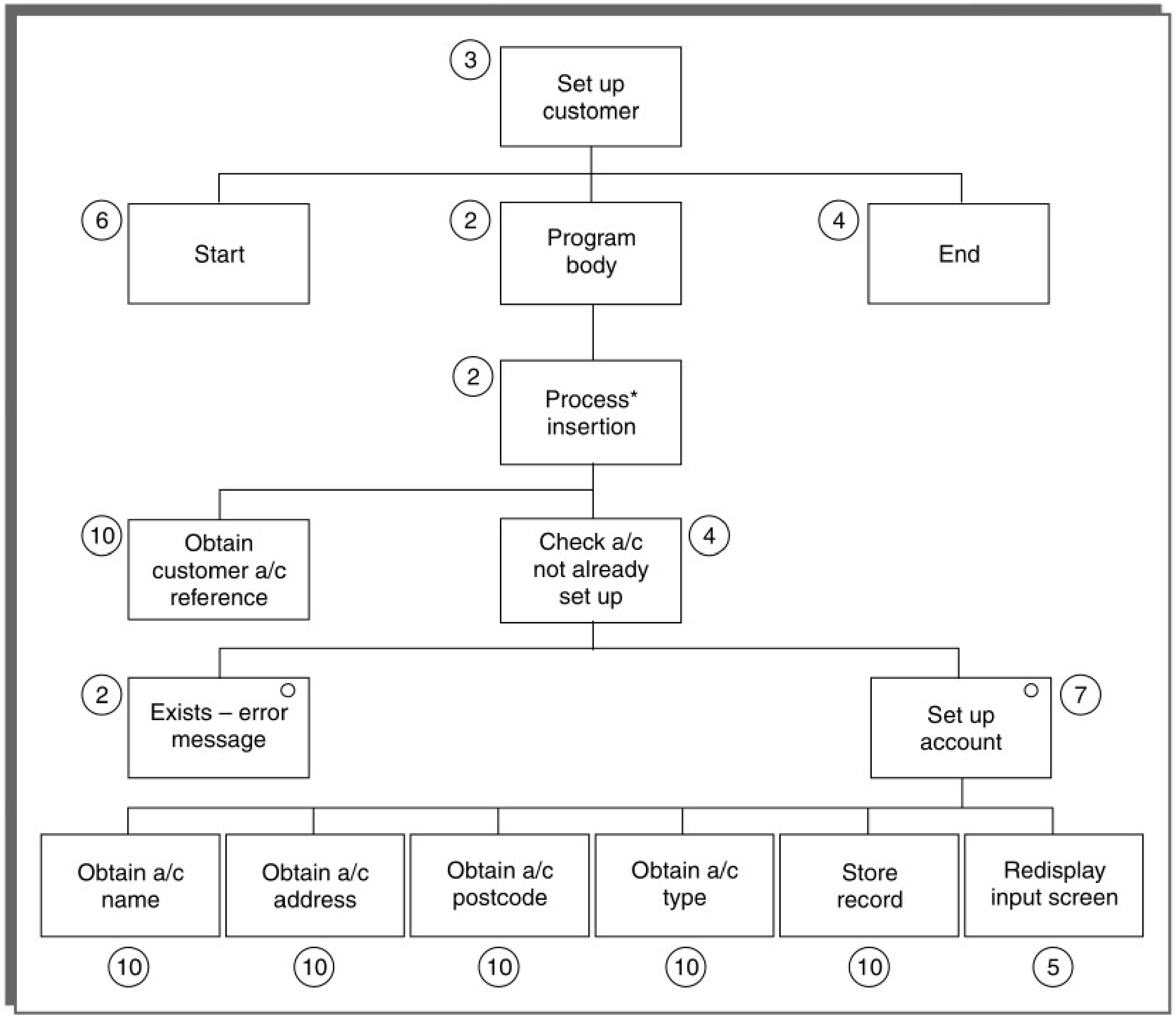
The operator will input:

Customer account number, Customer name, Address, Postcode, Customer type, Renewal date.

All this information will be set up in a CUSTOMER record on the system's database. If a CUSTOMER account already exists for the account number that has been input, an error message will be displayed to the operator.

Draw up an outline program structure diagram for a program to do the processing described above. For each box on your diagram, estimate the number of lines of code needed to implement the routine in a programming language that you are familiar with, such as Java.

Program Structure Diagram



Program Structure Diagram

Inventory Management System - Add New Product

In an inventory management system, there is a transaction that allows an operator to add new products into the system. The operator inputs the following details:

- Product ID
- Product Name
- Product Category
- Quantity
- Price
- Supplier ID

The system checks if the Product ID already exists. If it exists, an error message is displayed to the operator. If not, the product is added to the system’s database, and a success message is shown.

Program Structure Diagram

1-Main Program (InventoryManagementSystem)

This is the main entry point of the system that triggers the addition of a new product.

Responsibilities: Manages the process flow between components and user interactions.

Estimated LOC: 30-50 LOC

2-Input Module (ProductInputHandler)

Handles the collection of product information from the operator.

Responsibilities:

Input fields: Product ID, Product Name, Product Category, Quantity, Price, Supplier ID.

Validate if the fields are entered correctly (basic validation).

Estimated LOC: 60-80 LOC

Program Structure Diagram

3-Validation Module (ProductValidationHandler)

Validates the data provided by the operator.

Responsibilities:

Validate that the Product ID is unique.

Check that required fields (e.g., Product Name, Quantity) are provided.

Ensure valid data formats (e.g., Quantity is numeric, Price is positive).

Estimated LOC: 40-60 LOC

Program Structure Diagram

4- Product Lookup Module (ProductLookup)

Checks if the Product ID already exists in the inventory system.

Responsibilities:

Query the database to check for duplicate Product ID.

Return an error message if the Product ID already exists, or proceed if not found.

Estimated LOC: 40-60 LOC

Program Structure Diagram

5-Database Interaction Module (DatabaseHandler)

Handles database operations such as inserting new product records.

Responsibilities:

Establish a connection to the database.

Insert a new product into the products table.

Handle database connection errors and transaction management.

Estimated LOC:

Database Connection: 20-30 LOC

Insert Product Record: 40-60 LOC

Error Handling: 30-40 LOC

Total for Database Interaction: 90-130 LOC

Program Structure Diagram

6-Error Handling Module (ErrorHandler)

Handles errors related to invalid input, duplicate product IDs, or database issues.

Responsibilities:

Display user-friendly error messages when issues occur (e.g., "Product ID already exists").

Estimated LOC: 30-40 LOC

Program Structure Diagram

7-Success Message Module (SuccessHandler)

Displays a confirmation message to the operator when a product is successfully added to the inventory.

Responsibilities:

Show success message indicating that the product has been added to the system.

Estimated LOC: 20-30 LOC

Program Structure Diagram

8-Logging Module (LoggingHandler)

Logs every transaction, including successful product additions and errors, for auditing purposes.

Responsibilities:

Log details of product addition attempts.

Log any validation errors or system failures.

Estimated LOC: 20-30 LOC.

Top Down Approach

Top Down Approach

The Top-down approach to effort estimation begins with an overall understanding of the project and then breaks it down into smaller parts.

This method is based on historical data, expert judgment, and high-level requirements to make an overall estimate before breaking down into modules or tasks.

Key Points in the Top-down Approach:

Overall Estimation First:

The top-down approach starts with a high-level estimate based on past experience or analogous projects.

Breaking Down Components:

Once the total effort is estimated, the project is broken down into components or phases, and effort is allocated to each.

Contingency Adjustment:

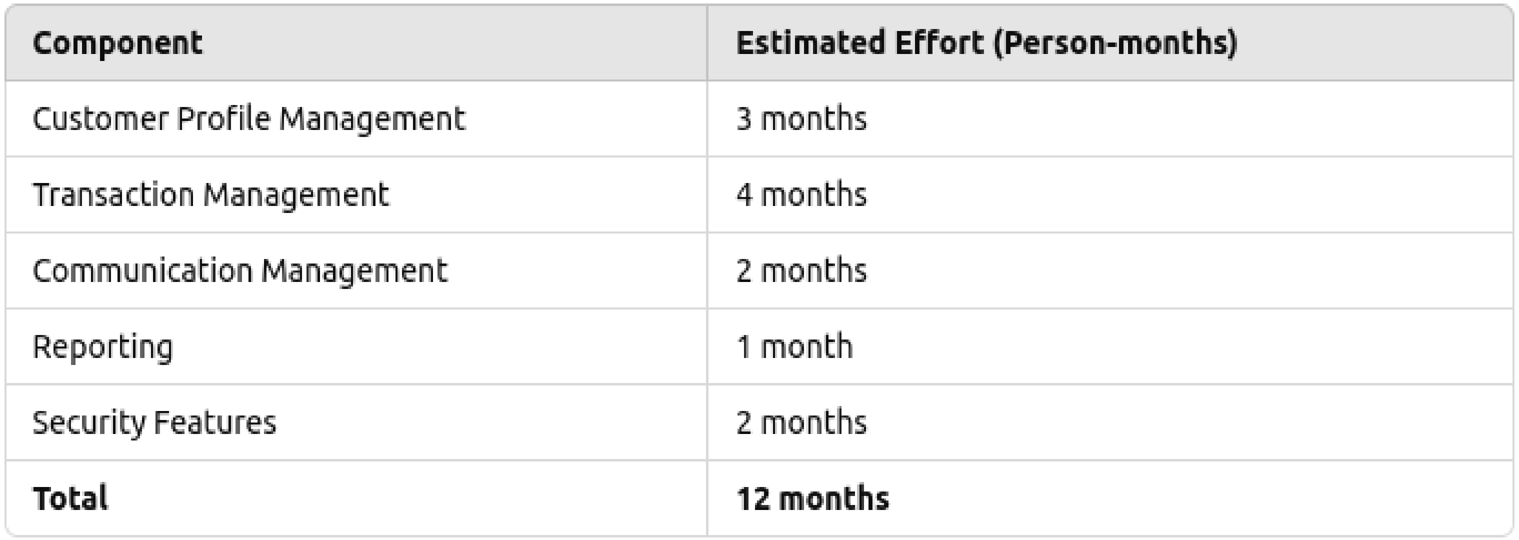
After estimating each component, a buffer is added to account for uncertainties or risks.

Top Down Approach

Example: Developing a CRM System

Let's assume that based on expert judgment and historical data from similar banking software, it is estimated that the CRM system will take 12 months to develop with 10 developers.

The team uses the top-down approach to refine the estimate by breaking the system into high-level components and allocating time and resources to each component.



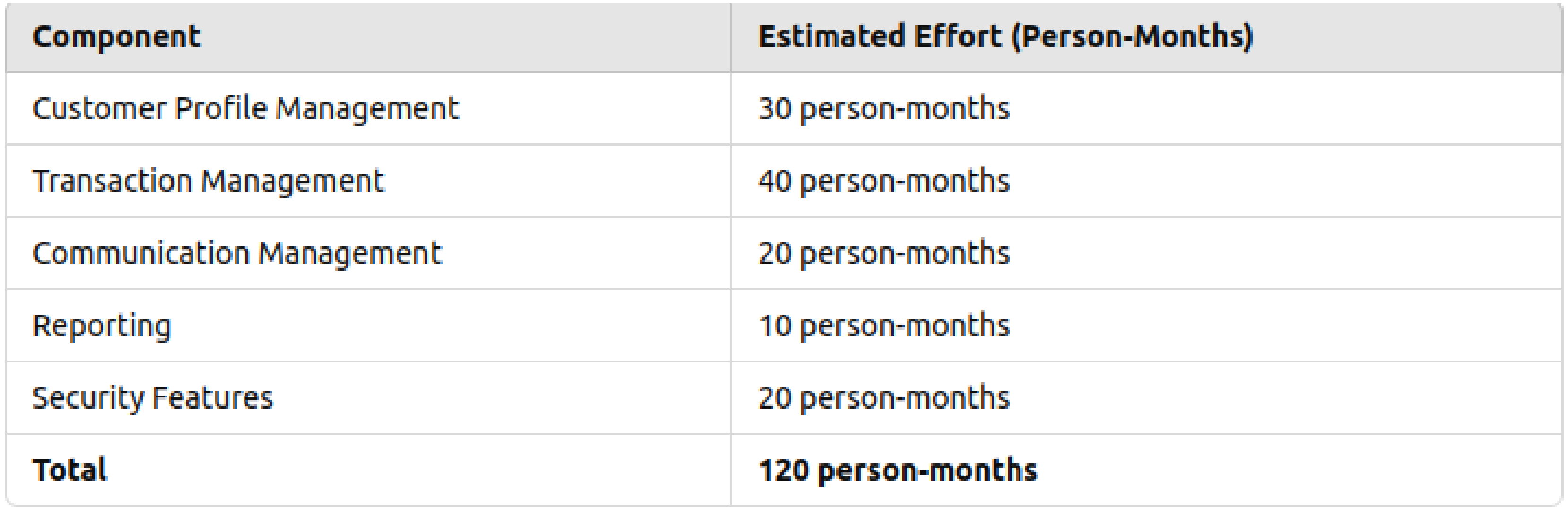
Component	Estimated Effort (Person-months)
Customer Profile Management	3 months
Transaction Management	4 months
Communication Management	2 months
Reporting	1 month
Security Features	2 months
Total	**12 months**

Top Down Approach

Overall Effort = 12 months×10 developers = 120 person-months.

Allocate Effort to Each Component based on:

- The complexity of the component.
- The size or functionality of each component relative to the entire system.
- Historical data or expert judgment about how long each component typically takes.



Component	Estimated Effort (Person-Months)
Customer Profile Management	30 person-months
Transaction Management	40 person-months
Communication Management	20 person-months
Reporting	10 person-months
Security Features	20 person-months
Total	**120 person-months**

Top Down Approach

Lets assume you decide to add 10% contingency to account for risks such as:

Integration with legacy banking systems.

Regulatory and security compliance requirements.

The contingency effort is calculated as:

Contingency Effort = 120 person-months × 10% = 12 person-months

Thus, the total adjusted effort becomes:

Total Effort = 120 person-months + 12 person-months = 132 person-months

Albrecht Function Point Analysis (FPA)

Albrecht Function Point Analysis (FPA)

Albrecht Function Point Analysis (FPA) is a structured technique for measuring the size and complexity of software systems based on their functional requirements.

It was introduced by Allan Albrecht in the late 1970s at IBM as a way to estimate the effort and resources needed for software development.

Instead of focusing on the lines of code, FPA evaluates the software based on the functionality it provides to its users.

Albrecht Function Point Analysis (FPA)

The fundamental idea behind **Function Point Analysis (FPA)** is that software is a collection of functions (inputs, outputs, inquiries, etc.), and the effort required to develop the software can be measured based on the complexity of these functions.

FPA helps in determining the size of the software in terms of function points (FPs), which can then be used to estimate project effort, cost, and duration.

Albrecht Function Point Analysis (FPA)

Components of FPA:

1. External Inputs (EI):
2. External Outputs (EO):
3. External Inquiries (EQ)
4. Internal Logical Files (ILF)
5. External Interface Files (EIF)

Albrecht Function Point Analysis (FPA)

1- External Inputs (EI):

These represent inputs into the system from external sources.

Examples include data entered by users, transactions submitted by other systems, and forms submitted via web interfaces.

Example:

Entering a customer’s details into a CRM system.

Albrecht Function Point Analysis (FPA)

2- External Outputs (EO):

These are outputs produced by the system that leave the system's boundary, such as reports, confirmations, and calculated results.

Example:

Generating a bank statement or transaction receipt.

Albrecht Function Point Analysis (FPA)

3- External Inquiries (EQ):

These are queries or retrievals of information from the system without any modifications to the data. It includes user-generated queries or lookups.

Example:

Retrieving a customer’s account balance.

Albrecht Function Point Analysis (FPA)

4- Internal Logic Files (ILF):

These are the logical data stores or databases maintained by the system. ILFs represent the internal data storage where the system maintains information.

Example:

A customer database or transaction history.

Albrecht Function Point Analysis (FPA)

5- External interface Files (EIF):

These are data stores that the system accesses but does not maintain, often belonging to other systems. An EIF is used to retrieve or query data from another system.

Example:

A bank’s CRM system accessing a credit bureau’s database for customer credit scores.

Albrecht Function Point Analysis (FPA)

Steps for FPA:

1. Count the Number of Each Function Type
2. Assign Complexity to Each Function
3. Apply the Weighting Factor
4. Calculate Unadjusted Function Points (UFP)
5. Adjust for Complexity (Value Adjustment Factor)
6. Calculate Final Adjusted Function Points

Albrecht Function Point Analysis (FPA)

Step-1. Count the Number of Each Function Type

For each of the five function types, count how many instances exist in the system.

For example, how many external inputs, external outputs, inquiries, internal logical files, and external interface files are present in the system.

Albrecht Function Point Analysis (FPA)

Step-2. Assign Complexity to Each Function

Each function type (EI, EO, EQ, ILF, EIF) is classified as Simple, Average, or Complex based on the number of data elements and the logical relationships involved.

This classification helps in determining the weighting factor for each function.

The classification is based on the number of **Data Element Types (DETs)** and **File Types Referenced (FTRs)** for inputs, outputs, and inquiries, or the **number of Record Element Types (RETs)** and **Data Element Types (DETs)** for files (ILFs and EIFs).

Albrecht Function Point Analysis (FPA)

Data Element Types (DETs): These are unique, user-recognizable fields (e.g., attributes like Customer Name, Customer ID, Address).

File Types Referenced (FTRs): Logical files accessed by a transaction (either ILFs or EIFs).

Record Element Types (RETs): Subgroups of data within an ILF or EIF (e.g., segments in a file like Customer Data and Transaction Data within a customer database).

Albrecht Function Point Analysis (FPA)

Guidelines for Assigning Complexity

1. External Inputs (EI)

An External Input (EI) is any input that updates data in the internal system (e.g., entering customer information, submitting a form).

Simple: Less than or equal to 4 DETs and 0 or 1 FTR.

Average: 5 to 15 DETs and 2 FTRs.

Complex: More than 15 DETs or 3 or more FTRs.

Example:

Simple: A form with three fields: Customer ID, Name, Address (3 DETs, 1 FTR - Customer Database).

Complex: A form with twenty fields that also checks three different files: Customer ID, Order History, Account Balance (20 DETs, 3 FTRs - Customer, Order, Account).

Albrecht Function Point Analysis (FPA)

2. External Outputs (EO)

An External Output (EO) represents the system-generated data provided to the user or another system (e.g., a report, invoice, or confirmation).

Simple: Less than or equal to 5 DETs and 0 or 1 FTR.

Average: 6 to 19 DETs and 2 or 3 FTRs.

Complex: More than 19 DETs or more than 3 FTRs.

Example:

Simple: A receipt with four fields: Transaction ID, Amount, Date, Customer Name (4 DETs, 1 FTR).

Complex: A detailed financial report pulling data from four different files and containing 25 fields (25 DETs, 4 FTRs).

Albrecht Function Point Analysis (FPA)

3. External Inquiries (EQ)

An External Inquiry (EQ) is a request to retrieve data without modifying it (e.g., viewing customer information, searching for a transaction).

Simple: Less than or equal to 5 DETs and 0 or 1 FTR.

Average: 6 to 19 DETs and 2 or 3 FTRs.

Complex: More than 19 DETs or more than 3 FTRs.

Example:

Simple: A search by Customer ID returning Name and Address (3 DETs, 1 FTR).

Complex: A query that retrieves customer history along with associated transactions, using multiple files (20 DETs, 3 FTRs).

Albrecht Function Point Analysis (FPA)

4. Internal Logical Files (ILF)

An Internal Logical File (ILF) is a logical grouping of data that the system maintains (e.g., customer database, transaction logs).

Simple: Less than or equal to 19 DETs and 1 RET.

Average: 20 to 50 DETs or 2 to 5 RETs.

Complex: More than 50 DETs or more than 5 RETs.

Example:

Simple: A customer database with one record type, storing basic data like ID, Name, Address (10 DETs, 1 RET).

Complex: A detailed customer file that stores several subgroups of information (contact details, transaction history, account preferences), having 6 subgroups and 100 fields (6 RETs, 100 DETs).

Albrecht Function Point Analysis (FPA)

5. External Interface Files (EIF)

An External Interface File (EIF) is a logical grouping of data maintained by an external system but accessed by the current system (e.g., a product catalog from another system).

Simple: Less than or equal to 19 DETs and 1 RET.

Average: 20 to 50 DETs or 2 to 5 RETs.

Complex: More than 50 DETs or more than 5 RETs.

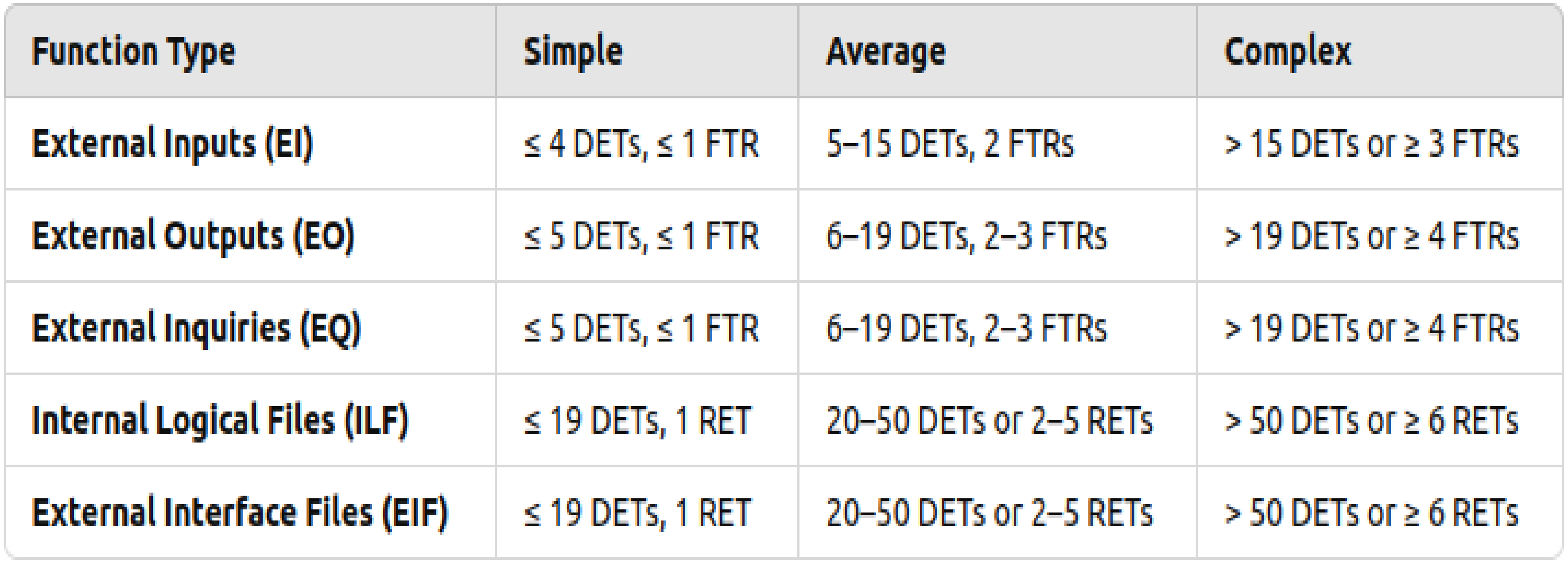
Example:

Simple: A reference to an external pricing file with a small number of fields and one logical grouping.

Complex: A detailed external product catalog with multiple groupings (product categories, price lists, stock levels) and hundreds of fields.

Albrecht Function Point Analysis (FPA)

Summary of Complexity Classification:

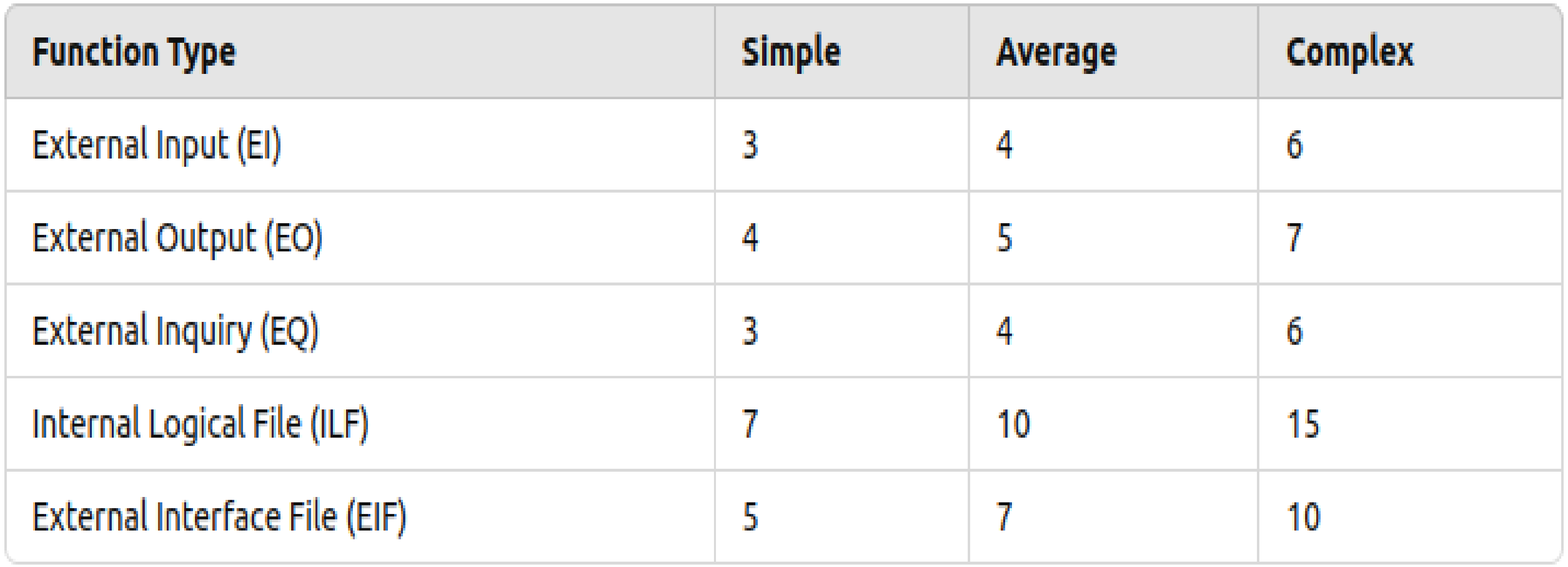


Function Type	Simple	Average	Complex
External Inputs (EI)	≤ 4 DETs, ≤ 1 FTR	5–15 DETs, 2 FTRs	> 15 DETs or ≥ 3 FTRs
External Outputs (EO)	≤ 5 DETs, ≤ 1 FTR	6–19 DETs, 2–3 FTRs	> 19 DETs or ≥ 4 FTRs
External Inquiries (EQ)	≤ 5 DETs, ≤ 1 FTR	6–19 DETs, 2–3 FTRs	> 19 DETs or ≥ 4 FTRs
Internal Logical Files (ILF)	≤ 19 DETs, 1 RET	20–50 DETs or 2–5 RETs	> 50 DETs or ≥ 6 RETs
External Interface Files (EIF)	≤ 19 DETs, 1 RET	20–50 DETs or 2–5 RETs	> 50 DETs or ≥ 6 RETs

Albrecht Function Point Analysis (FPA)

Step-3. Apply the Weighting Factor

Each function type has a specific weighting factor based on its complexity. Albrecht provided a table of standard weights to be used for each function type. The weights are shown below:



Function Type	Simple	Average	Complex
External Input (EI)	3	4	6
External Output (EO)	4	5	7
External Inquiry (EQ)	3	4	6
Internal Logical File (ILF)	7	10	15
External Interface File (EIF)	5	7	10

Albrecht Function Point Analysis (FPA)

Step-4. Calculate Unadjusted Function Points (UFP)

The Unadjusted Function Points (UFP) are calculated by multiplying the number of instances of each function type by its weighting factor and summing up the results.

$$\text{UFP}=\sum(\text{Number of Instances}\times\text{Weighting Factor})$$

For example, if a system has:

5 Simple External Inputs,, 3 Average External Outputs,, 2 Complex External Inquiries,

2 Average Internal Logical Files, 1 Complex External Interface File,

The calculation would be:

$$\text{UFP}=(5\times3)+(3\times5)+(2\times6)+(2\times10)+(1\times10)$$

$$\text{UFP}=15+15+12+20+10=72$$

Albrecht Function Point Analysis (FPA)

5. Adjust for Complexity (Value Adjustment Factor)

Once the unadjusted function points are calculated, they are adjusted for complexity based on 14 general system characteristics (GSCs). These characteristics account for the system's performance, usability, maintainability, and other environmental factors that affect development effort. The GSCs include factors like:

Each characteristic is rated on a scale from 0 to 5 (0 means no influence, 5 means strong influence), and the **Value Adjustment Factor (VAF)** is calculated using the formula:

$$VAF = 0.65 + (0.01 \times \text{Sum of the GSC ratings})$$

For example, if the sum of the GSC ratings is 40, the VAF would be:

$$VAF = 0.65 + (0.01 \times 40) = 0.65 + 0.4 = 1.05$$

Data communications

Distributed Data Processing

Performance considerations

Heavily used configuration

Transaction rate

Online data entry

End-user efficiency

Online updates

Complex processing

Reusability

Installation ease

Operational ease

Multiple sites

Facilitate change

Albrecht Function Point Analysis (FPA)

General System Characteristics:

1. Data Communications

Definition: This characteristic assesses how frequently and to what extent the system interacts with remote devices or other systems.

Examples: Systems with networked environments, multiple remote access points, or distributed databases.

Impact: The more complex the communication requirements, the higher the complexity rating. For example, systems that require constant communication over the internet or with remote sensors will score higher.

Albrecht Function Point Analysis (FPA)

General System Characteristics:

2. Distributed Data Processing

Definition: This evaluates whether the system handles data processing at multiple locations.

Examples: A system that performs data processing on distributed nodes or across a cloud infrastructure.

Impact: If the system involves distributed processing (e.g., cloud systems, microservices), the complexity increases.

Albrecht Function Point Analysis (FPA)

General System Characteristics:

3. Performance

Definition: This measures the need for system performance optimization, including response times, throughput, and transaction speed.

Examples: Real-time processing systems like stock trading platforms or gaming servers.

Impact: Systems requiring low-latency responses or high throughput will have higher performance requirements and thus score higher in complexity.

Albrecht Function Point Analysis (FPA)

General System Characteristics:

4. Heavily Used Configuration

Definition: This considers how much of the system is under constant or heavy use and what the performance demands are on that configuration.

Examples: Mission-critical systems, such as those for airlines or banks, which must handle heavy loads with minimal downtime.

Impact: Systems subject to high usage (e.g., thousands of concurrent users) will be rated as more complex.

Albrecht Function Point Analysis (FPA)

General System Characteristics:

5. Transaction Rate

Definition: This characteristic assesses how frequently transactions occur within the system.

Examples: High-frequency trading platforms, point-of-sale systems with thousands of transactions per minute.

Impact: The higher the transaction rate (number of transactions processed per second or minute), the greater the system’s complexity.

Albrecht Function Point Analysis (FPA)

General System Characteristics:

6. On-Line Data Entry

Definition: This evaluates the extent and complexity of real-time data entry in the system.

Examples: E-commerce websites, online banking, or ticket booking systems where users continuously input data.

Impact: Systems with extensive online data entry, especially real-time input validation, will score higher in complexity.

Albrecht Function Point Analysis (FPA)

General System Characteristics:

7. End-User Efficiency

Definition: This considers how user-friendly and efficient the system is for the end-users.

Examples: Systems with intuitive user interfaces, dashboards, or automation that reduce user effort.

Impact: The need for user-centric features, such as optimized navigation and ease of use, increases complexity, especially if the system must support a wide range of users with varying technical skills.

Albrecht Function Point Analysis (FPA)

General System Characteristics:

8. On-Line Update

Definition: This characteristic measures the extent to which the system allows for updates in real-time.

Examples: A system that allows users to modify customer records in real-time or change pricing information in an e-commerce system.

Impact: Systems that require real-time updates to multiple data points will have higher complexity due to the need for synchronization and error handling.

Albrecht Function Point Analysis (FPA)

General System Characteristics:

9. Complex Processing

Definition: This assesses the complexity of internal business logic, including algorithms, calculations, and data transformations.

Examples: Inventory management systems with complex re-ordering rules or financial systems with sophisticated tax calculations.

Impact: The more complex the business rules, calculations, or workflows, the higher the complexity rating.

Albrecht Function Point Analysis (FPA)

General System Characteristics:

10. Reusability

Definition: This characteristic looks at how much of the system’s components are designed to be reusable in other applications or systems.

Examples: Modular software systems, libraries, or APIs designed for reusability across different projects.

Impact: The higher the level of reusability of code or components, the greater the complexity due to the need for generalization and abstraction.

Albrecht Function Point Analysis (FPA)

General System Characteristics:

11. Installation Ease

Definition: This measures how easy or complex it is to install the system in the production environment.

Examples: Systems that require extensive configuration, data migration, or compatibility checks during installation.

Impact: If the system has many dependencies or requires extensive setup, it will score higher on complexity. Systems with simple installations, like plug-and-play applications, score lower.

Albrecht Function Point Analysis (FPA)

General System Characteristics:

12. Operational Ease

Definition: This considers how simple it is for operators to run and maintain the system.

Examples: Systems with intuitive control panels, automated backups, and straightforward error-handling mechanisms.

Impact: If the system includes tools that simplify operations (e.g., automation, system health monitoring), it will reduce the complexity. However, systems requiring extensive operator intervention will be rated as more complex

Albrecht Function Point Analysis (FPA)

General System Characteristics:

13. Multiple Sites

Definition: This characteristic evaluates how the system supports deployment and usage across multiple physical locations.

Examples: Retail POS systems used across different store locations or ERP systems deployed in different branches of an organization.

Impact: Systems deployed across multiple locations must handle issues like data synchronization, network reliability, and different local configurations, increasing complexity.

Albrecht Function Point Analysis (FPA)

General System Characteristics:

14. Facilitate Change

Definition: This measures how easily the system can be changed or enhanced to meet evolving business needs.

Examples: Systems designed with flexibility in mind, such as those with modular architecture or those that follow Agile development principles.

Impact: Systems designed for flexibility and future adaptability (like those using microservices or modular components) are more complex to develop but easier to modify, while rigid systems score lower.

Albrecht Function Point Analysis (FPA)

6. Calculate Final Adjusted Function Points

Finally, the adjusted function points (AFP) are calculated by multiplying the unadjusted function points (UFP) by the value adjustment factor (VAF):

$$\text{AFP} = \text{UFP} \times \text{VAF}$$

$$\text{AFP} = 72 \times 1.05 = 75.6$$

Albrecht Function Point Analysis (FPA)

How Function Points Are Used

Once you have calculated the number of function points, you can use them to:

Estimate Effort: Convert function points into person-hours or person-months by applying productivity rates (e.g., 10 function points per person-month).

Estimate Cost: Based on the effort, you can estimate the project cost using standard rates (e.g., cost per person-month).

Estimate Time: By knowing the effort and team size, you can estimate the project duration.

Compare Projects: Since function points are technology-agnostic, they can be used to compare the size and complexity of different projects regardless of the programming language or tools used.

Case Study (FPA)

Customer Relationship Management System

A financial institution wants to implement a ***Customer Relationship Management (CRM)*** system to streamline customer interactions, manage client information, and improve service delivery. The system will handle customer profiles, transactions, communications, and reporting, along with strong security measures.

The institution hires a software development team to estimate the effort, cost, and time required to build this system.

The team uses ***Function Point Analysis (FPA)*** to measure the size of the system and estimate the development effort.

Customer Relationship Management System

Step 1: Identifying Functions in the CRM System

The first step in Function Point Analysis (FPA) for the CRM system is to identify the types of functions in the system.

These include:

a. External Inputs (EI)

Functions where data is input into the system by the user or an external system.

Example in CRM:

Adding new customer information (e.g., name, address, contact details).

Updating customer information (e.g., changing phone number or address).

In function point terms, each data input screen is an External Input.

Customer Relationship Management System

Step 1: Identifying Functions in the CRM System

b. External Outputs (EO)

Functions where the system produces calculated or processed data for external use.

Example in CRM:

Generating customer reports, invoices, or summaries.

Outputting data for analysis or alerts, like overdue payments.

In function point terms, each report generated by the system is an External Output.

Customer Relationship Management System

Step 1: Identifying Functions in the CRM System

c. External Inquiries (EQ)

Simple data retrieval functions that display information without complex processing.

Example in CRM:

Searching for a customer’s contact details or their transaction history.

Querying the system to view information like sales records.

In function point terms, each query-based retrieval is an External Inquiry

Customer Relationship Management System

Step 1: Identifying Functions in the CRM System

d. Internal Logical Files (ILF)

Logical files that the system maintains and controls. These files are critical data stores within the system.

Example in CRM:

The Customer Database, which stores all customer-related information.

The Transaction Log, which keeps a record of customer purchases and interactions.

In function point terms, each major logical data store is an Internal Logical File.

Customer Relationship Management System

Step 1: Identifying Functions in the CRM System

e. External Interface Files (EIF)

Logical files that the system does not maintain but accesses from external systems.

Example in CRM:

Accessing product data from another system for invoicing purposes.

Accessing a third-party marketing database.

In function point terms, each external file the system interfaces with is an External Interface File.

Customer Relationship Management System

Step 2: Estimating the Function Point Count (Assigning Complexity)

Now that we have identified the functions, we can move to Step 2, where we calculate the function points based on the number of Data Element Types (DETs) and File Types Referenced (FTRs).

1. External Inputs (EI)

Function: Add new customer data (e.g., Name, Address, Phone Number, Email).

Data Elements (DETs): 4 fields (Name, Address, Phone Number, Email).

File Types Referenced (FTRs): 1 file (Customer Database).

Complexity: This is a Low Complexity EI (4 DETs and 1 FTR).

Function Point Count: For low complexity, the standard function point count is 3.

Function: Update customer details (similar DETs and FTRs as above).

Function Point Count: 3 (Low Complexity).

Customer Relationship Management System

Step 2: Estimating the Function Point Count (Assigning Complexity)

2. External Outputs (EO)

Function: Generate a customer invoice (involves summarizing purchases, taxes, and discounts).

Data Elements (DETs): 5 fields (Customer Name, Total Purchases, Discount, Tax, Final Amount).

File Types Referenced (FTRs): 2 files (Customer Database, Transaction Log).

Complexity: This is a Low Complexity EO (5 DETs and 2 FTRs).

Function Point Count: For low complexity, the standard function point count is 4.

Function: Generate customer transaction report (summary of purchase history).

Function Point Count: 4 (Low Complexity).

Customer Relationship Management System

Step 2: Estimating the Function Point Count (Assigning Complexity)

3. External Inquiries (EQ)

Function: Query customer details (e.g., search for customer by name).

Data Elements (DETs): 3 fields (Customer Name, Address, Phone Number).

File Types Referenced (FTRs): 1 file (Customer Database).

Complexity: This is a Low Complexity EQ (3 DETs and 1 FTR).

Function Point Count: 3.

Function: Query transaction history.

Function Point Count: 3 (Low Complexity).

Customer Relationship Management System

Step 2: Estimating the Function Point Count (Assigning Complexity)

4. Internal Logical Files (ILF)

Function: Maintain the customer database.

Data Elements (DETs): 4 fields (Customer Name, Address, Phone Number, Email).

Complexity: Based on its size (number of fields), this is considered a Low Complexity ILF.

Function Point Count: For low complexity, the standard function point count is 7.

Function: Maintain the transaction log.

Function Point Count: 7 (Low Complexity).

Customer Relationship Management System

Step 2: Estimating the Function Point Count (Assigning Complexity)

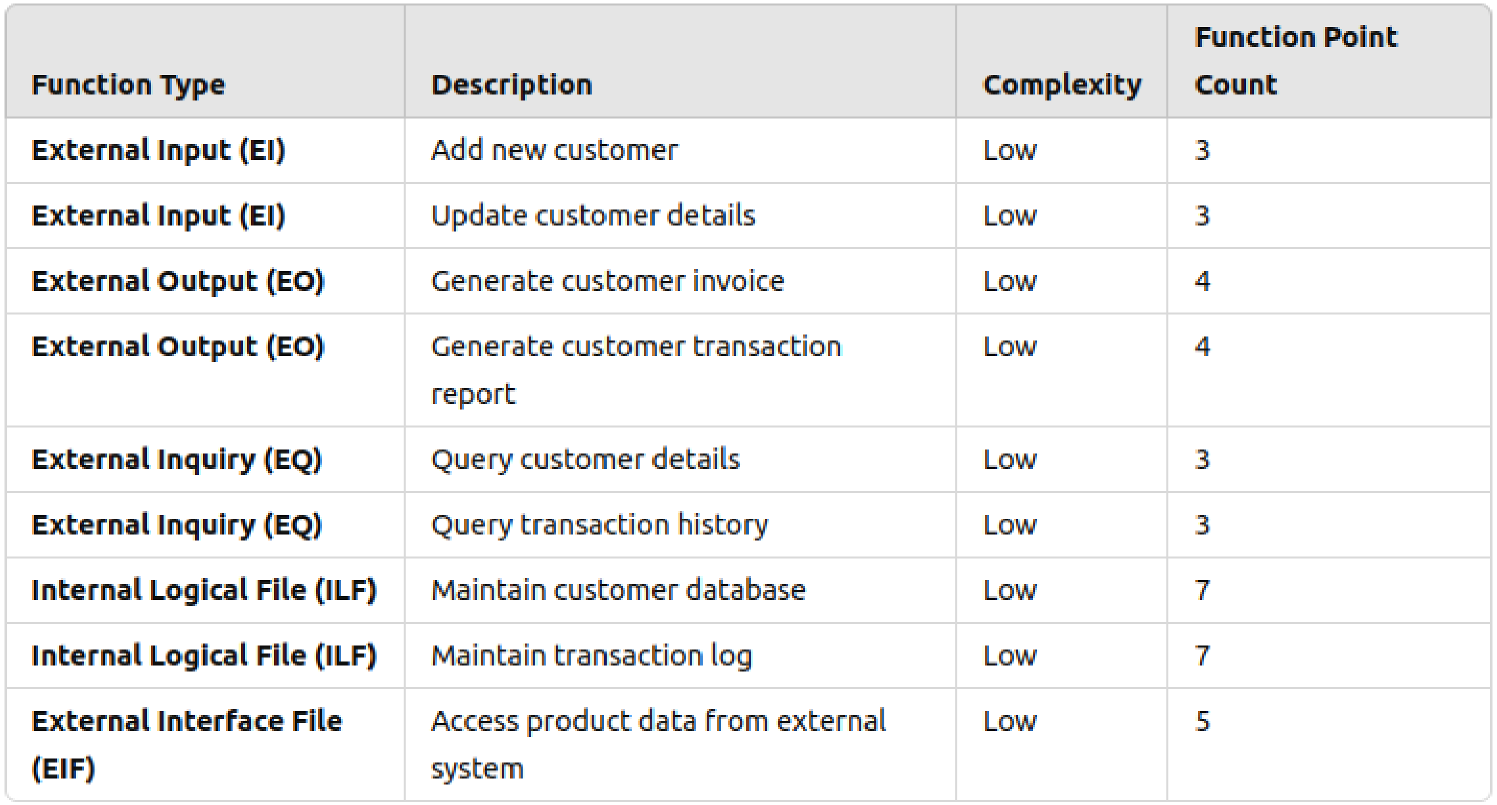
5. External Interface Files (EIF)

Function: Access product data from external system.

Complexity: This is a Low Complexity EIF.

Function Point Count: 5.

Customer Relationship Management System

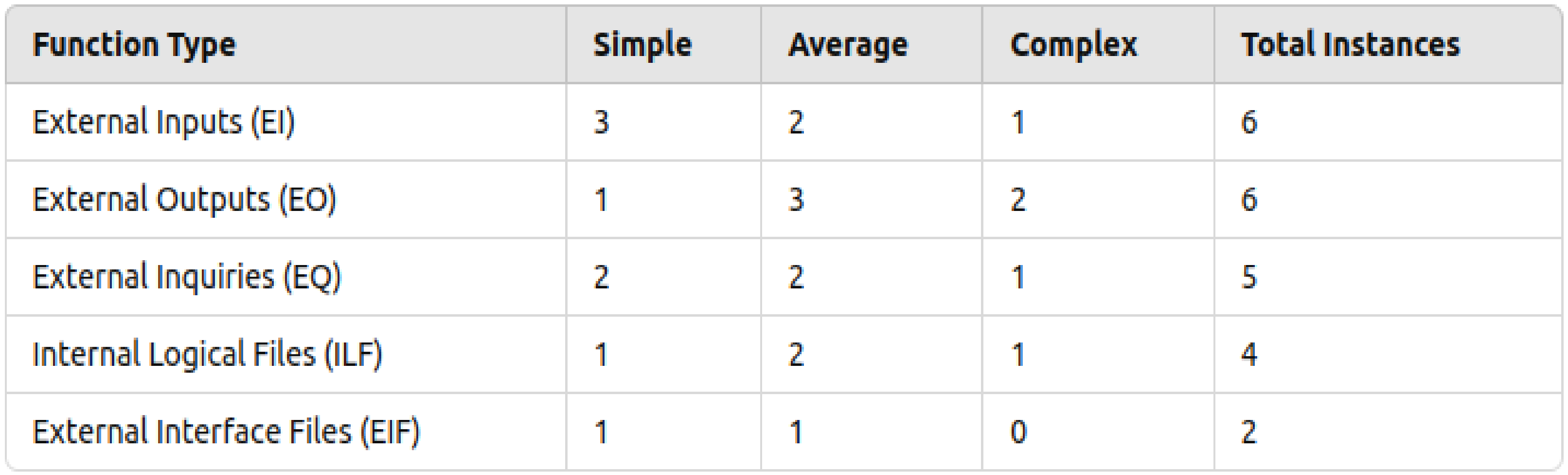


Function Type	Description	Complexity	Function Point Count
External Input (EI)	Add new customer	Low	3
External Input (EI)	Update customer details	Low	3
External Output (EO)	Generate customer invoice	Low	4
External Output (EO)	Generate customer transaction report	Low	4
External Inquiry (EQ)	Query customer details	Low	3
External Inquiry (EQ)	Query transaction history	Low	3
Internal Logical File (ILF)	Maintain customer database	Low	7
Internal Logical File (ILF)	Maintain transaction log	Low	7
External Interface File (EIF)	Access product data from external system	Low	5

3+3+4+4+3+3+7+7+5=39 function points

Customer Relationship Management System

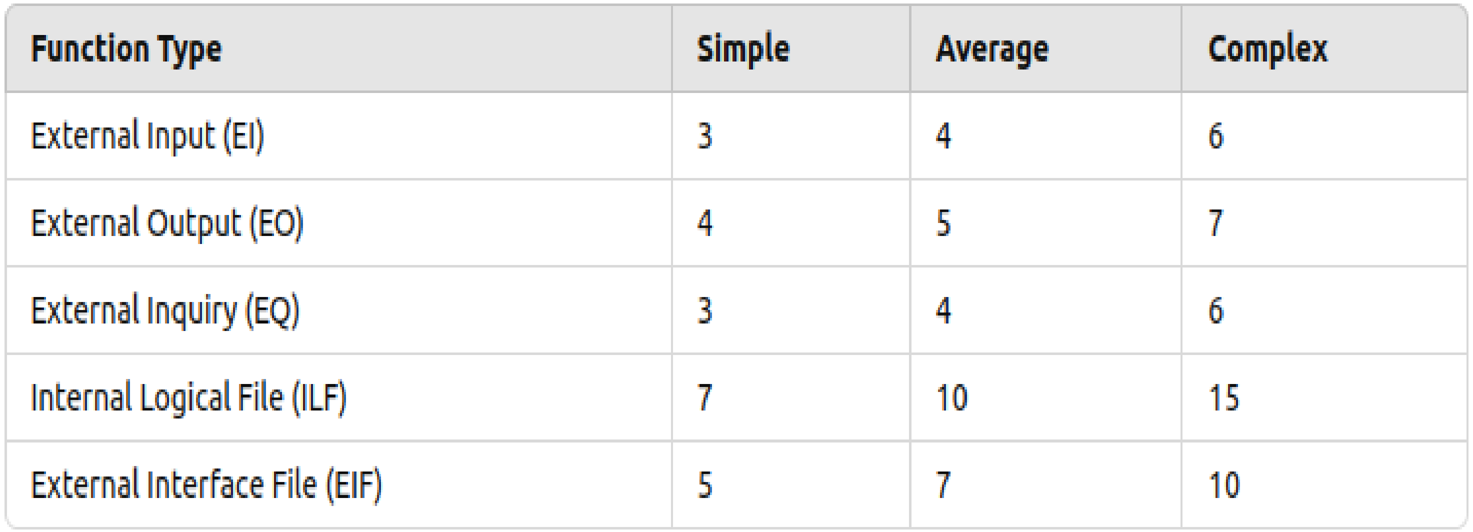
Lets Consider another table.



Function Type	Simple	Average	Complex	Total Instances
External Inputs (EI)	3	2	1	6
External Outputs (EO)	1	3	2	6
External Inquiries (EQ)	2	2	1	5
Internal Logical Files (ILF)	1	2	1	4
External Interface Files (EIF)	1	1	0	2

Customer Relationship Management System

Step 3: Apply the Weighting Factor



Function Type	Simple	Average	Complex
External Input (EI)	3	4	6
External Output (EO)	4	5	7
External Inquiry (EQ)	3	4	6
Internal Logical File (ILF)	7	10	15
External Interface File (EIF)	5	7	10

Customer Relationship Management System

Step 4: Calculate Unadjusted Function Points (UFP)

External Inputs (EI):

$(3\times3)+(2\times4)+(1\times6)=9+8+6=23$ FP

External Outputs (EO)

$(1\times4)+(3\times5)+(2\times7)=4+15+14=33$ FP

External Inquiries (EQ):

$(2\times3)+(2\times4)+(1\times6)=6+8+6=20$ FP

Internal Logical Files (ILF):

$(1\times7)+(2\times10)+(1\times15)=7+20+15=42$ FP

External Interface Files (EIF):

$(1\times5)+(1\times7)=5+7=12$ FP

$$\textbf{UFP}=23+33+20+42+12=130$$

Customer Relationship Management System

Step 5: Apply the Value Adjustment Factor (VAF)

Suppose the sum of GSC’s ratings is 45.

The Value Adjustment Factor (VAF) is calculated as:

$$\mathbf{VAF} = 0.65 + (0.01 \times 45) = 0.65 + 0.45 = 1.10$$

Customer Relationship Management System

Step 6: Calculate Adjusted Function Points (AFP)

$$AFP = UFP \times VAF$$

$$AFP = 130 \times 1.10 = 143\ FP$$

Customer Relationship Management System

Step 7: Estimating Effort, Cost, and Time

Effort Estimation

Typically, effort is estimated in person-hours or person-months. Suppose the organization has a historical productivity rate of 8 function points per person-month.

Effort (person-months) = AFP / Productivity rate **= 143 / 8 = 17.88 person-months**

Cost Estimation

Assuming the cost of one person-month is 10,000, the total cost of the project would be:

Cost=18 person-months×10,000=180,000

Time Estimation

If the development team consists of 3 full-time developers, the time to complete the project would be:

Time (months) = Effort (person-months)

Number of developers =

18 / 3 = 6 months