Measuring Internal Product Attributes: Software Size

Function Points

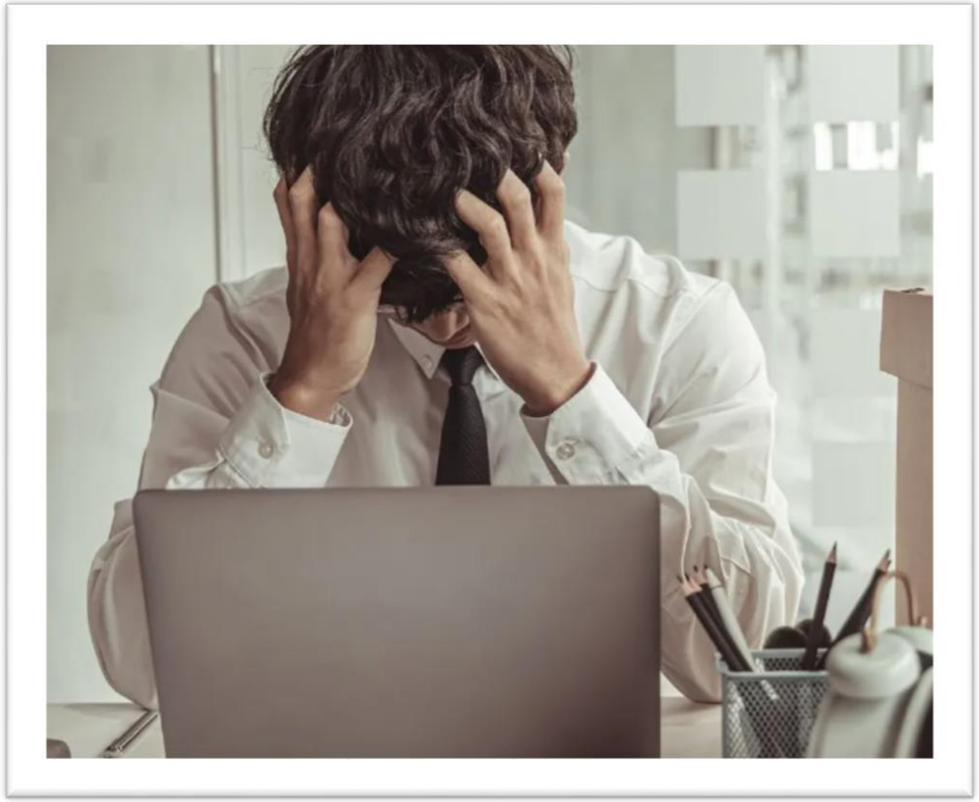
Lecturer Abdullah Al Jobair (CSE, UIU)

Honoring the Socratic Paradox

“I know that I know nothing at all.”

- Socrates

It is quite common for any developer to feel uncomfortable when requested to estimate a development project that they know absolutely nothing about !!!



The trick is to break the project down into smaller chunks that are recognizable, and then estimate these chunks. Then you simply aggregate the estimates for all these chunks together to arrive at an estimate for the project as a whole. Sound simple? It is, using Function Points as a basis for identifying the chunks! Before deep diving into Function Point Analysis let's recap few things...

Software Size

- Size measurement must reflect effort, cost and productivity.
- Size-oriented metrics are direct measures of software and the process by which it was developed.
- These metrics include effort (time), money spent, LOC, pages of documents created, errors, and number of staff.
- Basic measure for length is LOC. Simple size-oriented metrics can be generated from LOC, such as:
  - Productivity = KLOC / person-month
  - Quality = defects / KLOC
  - Documentation = pages of documents / KLOC

Function-Oriented Metrics

- Function Point Analysis (FPA) is a weighted measure of software functionality.
- The idea is that a product with more functionality will be larger in size.
- Function-oriented metrics are indirect measures of software which focus on functionality and utility.
- The first function-oriented metrics was proposed by Albrecht (1979~1983) who suggested a productivity measurement approach called the Function Point(FP) method.
- FP maintained by IFPUG (International Function Point Users Group)

Function-Oriented Metrics

Size Metric: **FUNCTION POINT ANALYSIS (FPA)**

- ✓ It is based on the idea that the **software size should be measured according to the functionalities specified by the user.**
- ✓ Therefore, FPA is a standardized methodology for **measuring various functions** of a software **from the user's point of view.**
- ✓ **The size of an application is measured in function points**.

Function Points, at their simplest level of understanding, fall into 5 categories:

1. Inputs,
2. Outputs,
3. Queries,
4. Internal Files and,
5. External Files

Well that seems simple enough, right? Well that’s not so simple as it sounds. because, in the real world, there are always gray areas.

Keep It Simple, Stupid - Principle

Keep it simple, stupid is a design principle which states that designs and/or systems should be as simple as possible.

The K.I.S.S principle should be applied here. My recommendation is to follow your instincts and allow these simple definitions to guide your identification and categorization of the function points. So here goes:

Inputs – This is **how an application gets information**. The most obvious example is a **data input field on a screen.**

Outputs – **Information that leaves the application, whether by report, file or screen**. Information that leaves the application **can be used by another application or by the user**.

Queries – **A process by which both inputs and outputs are utilized in data retrieval from one or more Internal or External Files**. Ah, but **we haven't yet defined what internal and external files are** !!!!!

Internal Files – a **group of logically related data that is contained within the system** that you are estimating AND is **maintained by one or more Inputs to the system**.

External Files – a **group of logically related data that is not contained within the system** you are estimating and/or is **maintained by another applications inputs**. External files are for reference purposes only within the bounds of the system you are estimating. An example would be an Address Book application that provides an extract of employee addresses (via an external file) to an HR application.

Size Metric: FUNCTION POINT ANALYSIS (FPA)

✓ In counting FPs there are five standard "functions" that you count. The 5 types of EP’s can be grouped into 2 types of functions: The first two of these are called Data Functions, and last three are called Transaction Functions.

✓ **Two data function types:**

- **Internal Logical Files (ILF):** these files are **the master or transaction files** that the **system interacts with during its session.**
- **External Interface Files(EIF):** represent the **data** that your **application will use/reference**, but **data that is**

✓ **Three Transactional function types:**

- **External Inputs (EI)**: these are **end-user actions** such as putting in a **login or executing a mouse click**.
- **External Outputs (EO):** the system provides the **end-user output or interface** such as a **GUI display or items in a report.**
- **External Inquiries (EQ):** this function is **initiated by the end-user**. For example, the **end-user wishes to submit a query** to a database or requests on-line help. In any case the **developer provides a means for the end-user to "search" for answers.**

Using this terminology, when a person that counts FPs looks at a software system, they see something like this:

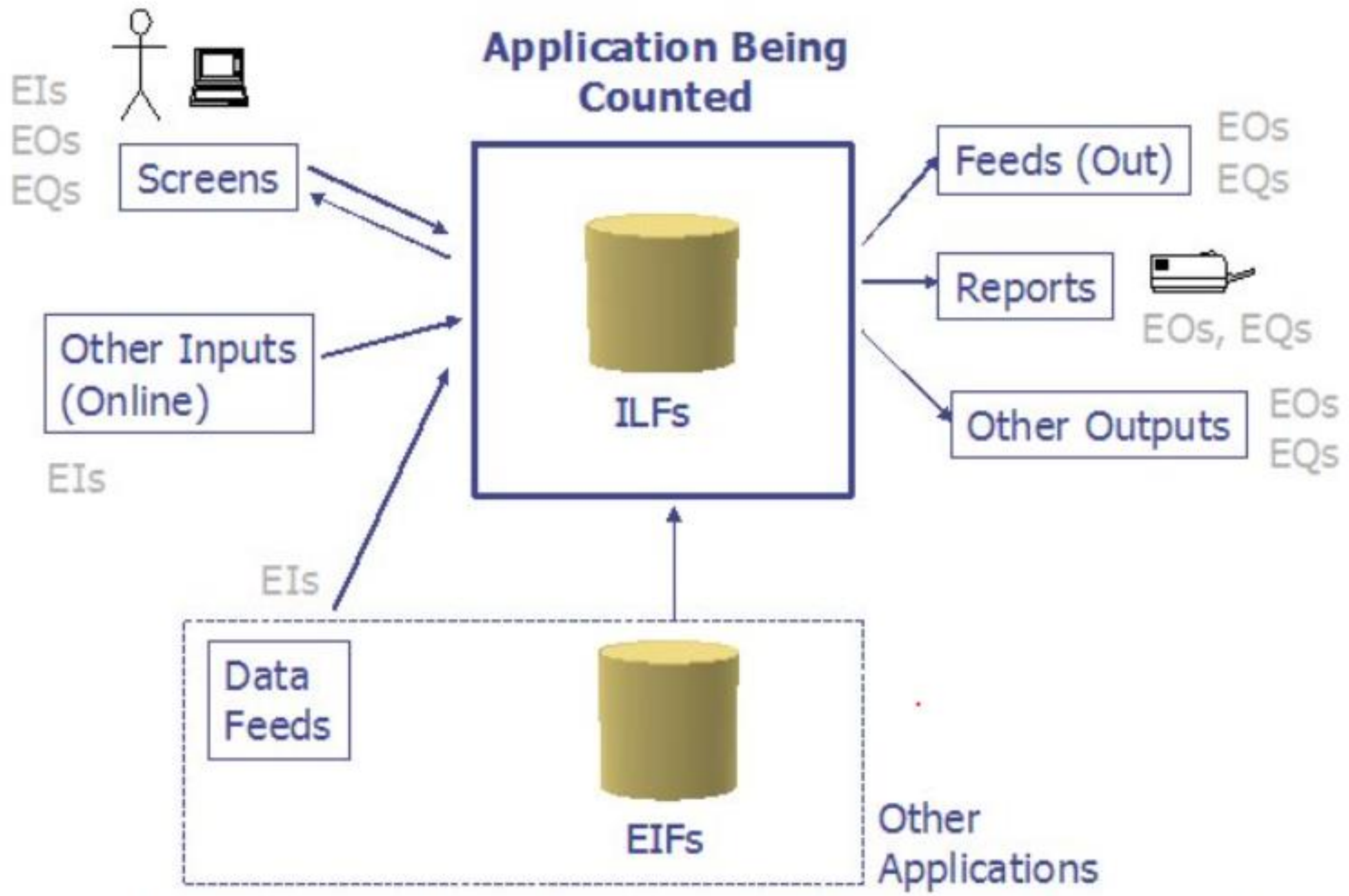


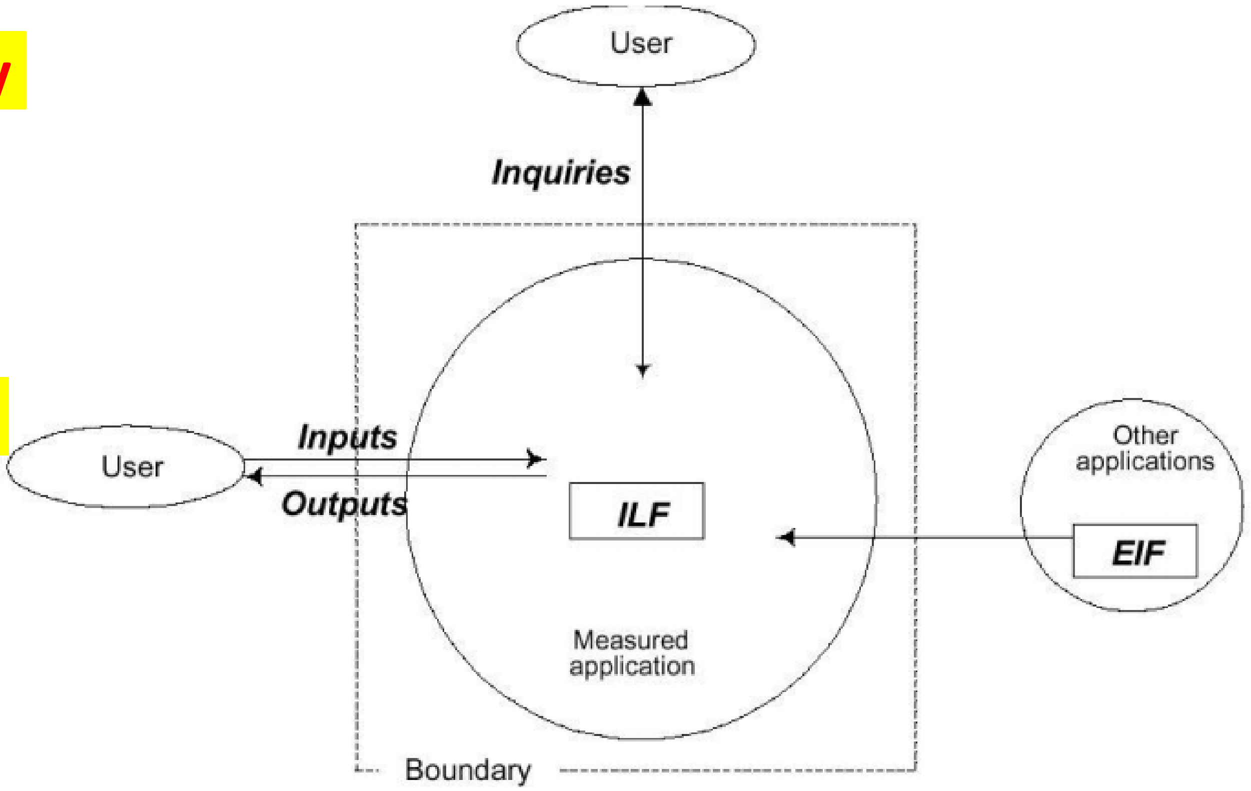
Figure 1: The view of a software application from the eyes of a Function Point practitioner.

These five functions will be discussed in greater depth in the sections that follow.

Function-Oriented Metrics

Size Metric: **FUNCTION POINT ANALYSIS (FPA)**

- ✓ **The first step in calculating FP is to identify the counting boundary.**
- ✓ **Counting boundary:** The **border between the application or project being measured and external applications or the user domain.**
- ✓ **A boundary establishes which functions are included in the function point count**



Before getting into the details of the five functions there are several terms that you need to understand, because they'll be used in each of the subsequent definitions.

Important terms and definitions used in describing the five functions:

- **User identifiable**
- **Control information**
- **Elementary process**
- **Data Element Type, or DET**
- **Record Element Type, or RET**
- **File Type Referenced, or FTR**

Let's use a real-world example to illustrate these concepts: Example: Library Management System

User identifiable

This term refers to defined requirements for processes and/or groups of data that are agreed upon, and understood by, both the users and software developers.

Explanation: It means that there are clear and agreed-upon requirements for processes (tasks) or groups of data in the software. Both the users (people who will use the software) and the developers (people building the software) understand and agree on these requirements.

Imagine you're building a Library Management System. User identifiable means that there are clear requirements that both librarians (users) and developers understand. For instance, **the system should allow librarians to add books, check out books to users, and manage the library catalog.**

Example: **Library Management System**

Control information

This is data that influences and elementary process of the application being counted. It specifies what, when, or how data is to be processed.

Explanation: This is data that guides **how a specific action or process in the software should happen. It tells the software what to do, when to do it, or how to process data**.

In our Library Management System, control information would be instructions on how to process certain actions. For instance, **when a librarian checks out a book, the control information specifies how to update the system to reflect that the book is now borrowed.**

Example: Library Management System

Elementary process

An elementary process is the smallest unit of activity that is meaningful to the user. An elementary process must be self-contained and leave the business of the application being counted in a consistent state.

In our system, checking in a returned book could be an elementary process. It's a self-contained action that, when done, ensures that the library catalog is in a consistent state.

Example: Library Management System

Data Element Type, or DET

A data element type is a unique, user recognizable, non-repeated field. **This definition applies to both analyses of data functions and transactional functions.**

Explanation: For example, if you're talking about a list of students, each student's name or ID would be a DET.

Let's consider the data about a book. Each piece of information like the **book's title, author, and ISBN is a DET.**
For example:
Book Title (DET): "To Kill a Mockingbird"
Author (DET): "Harper Lee"
ISBN (DET): “32axxxxx42948290"

Example: Library Management System

Record Element Type, or RET

A record element type is a user recognizable **subgroup of data elements** **within an Internal Logical File or External Interface File.**

Explanation: A RET is a **group of related data elements within a file.** If you're dealing with a file that contains information about students, a **single student's record (with their name, ID, and other details) would be a RET.**

Now, think of the **entire record for a book**. This is a **group of related data elements forming a larger unit—a RET**. For example

Record for a Book (RET):

Book Title (DET): "To Kill a Mockingbird"

Author (DET): "Harper Lee"

ISBN (DET): "9780061120084"

Example: Library Management System

File Type Reference, or FTR

It is a term used to describe a situation where a certain file type is assessed or referenced by a transaction in a software application. A **transaction** refers to an elementary process, which is the smallest unit of activity that is meaningful to the user

Explanation-

Transaction: Borrowing a book from the library.

File Type Referenced (FTR):

- The FTR in this case could be an Internal Logical File (ILF) representing the database of available books within the library. The transaction of borrowing a book references this internal file.

So, in summary, when you see "FTR – File Type Referenced," it means that a transaction in the software is interacting with a specific type of file, and that file can be either an Internal Logical File (ILF) or an External Interface File (EIF).

Quick Recap:
There are 5 different types of function points, also known as Elementary Processes (EP's) – Inputs, outputs, queries, internal files and external files. The 5 types of EP's can be grouped into 2 types of functions:

Inputs, Outputs & Queries all qualify as Transactional Functions and,
Internal Files & External Files are distinguished as Data Functions

A Transactional Function is broken down into DET's and FTR's, while a Data Function is broken down into DET's and RET's.

In simpler terms, imagine you have a box of LEGO pieces:

Each LEGO piece is like a DET—a single, unique part.
When you put several LEGO pieces together to build a small car or a person, that whole structure is like a RET—a group of related pieces forming a larger unit.

So, in the context of software development, DETs are the individual pieces of information, and RETs are groups of these pieces that make up a meaningful unit within the software.

In this Library Management System:

The user identifiable aspect includes agreed-upon functionalities between librarians and developers.
Control information guides how processes like checking out books should be handled.
Elementary processes are the smallest meaningful actions, like checking in a returned book.
DETs are individual pieces of data, such as the title, author, and ISBN of a book.
RETs are groups of related DETs, forming a complete record for a book.

Understanding these terms helps in analyzing and measuring the functionality of the software system using techniques like Function Point Analysis.

Function Point (FP) Count

- Function points (FPs) measure the amount of functionality in a system based upon the system specification.
- FP relies on 5 things:
  - *External inputs (EI)*
  - *External outputs (EO)*
  - *External inquiries (EQ)*
  - *External Interface Files (EIF)*
  - *Internal Logical Files (ILF)*
- FP is computed in two steps:
  - Calculating Unadjusted Function point Count (UFC).
  - Multiplying the UFC by a Technical complexity factors (TCF)

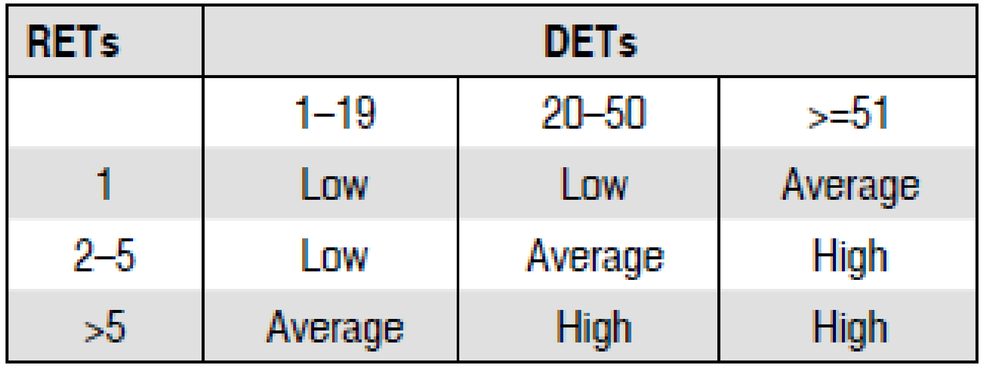
  The final (adjusted) Function Point is: FP = UFC * TCF

SW Metrics

Size Metric: **FUNCTION POINT ANALYSIS (FPA)**

✓ **Data Function complexity:**

- ✓ Each identified data function must be **assigned a functional complexity** based on the number of **data element types (DETs)** and **record element types (RETs)** associated with the ILF and EIF.
- ✓ DETs are **unique user-recognizable**, ***non-repeatable fields or attributes.***
- ✓ RETs are **user-recognizable subgroups** (optional or mandatory) of **data elements contained within an ILF or EIF.** Subgroups are typically represented in an entity relationship diagram as entity subtypes or attribute entities.
- ✓ **Functional complexity is the rating assigned to each data function**.



RETs	DETs		
	1–19	20–50	>=51
1	Low	Low	Average
2–5	Low	Average	High
>5	Average	High	High

For developing a Library Management System (LMS) for UIU. The system needs to handle various functions such as managing books, users, and transactions. The system maintains 3 types of file for i) storing information about books ii) storing user information iii) recording transactions.

Let's break down the scenario and identify Data Element Types (DET) and Record Element Types (RET):

Internal Logical Files (ILFs):

1.File for Storing Information about Books:

1. DET: Book ID, Title, Author, Genre, ISBN, Publication Year, Number of Copies, etc.
2. RET: Each record representing a unique book with the associated data elements.

2.File for Storing User Information:

1. DET: User ID, Name, Address, Email, Phone Number, Membership Status, etc.
2. RET: Each record representing a unique user with the associated data elements.

3.File for Recording Transactions:

1. DET: Transaction ID, Book ID, User ID, Transaction Date, Due Date, Status, etc.
2. RET: Each record representing a unique transaction with the associated data elements.

Function Point (FP)

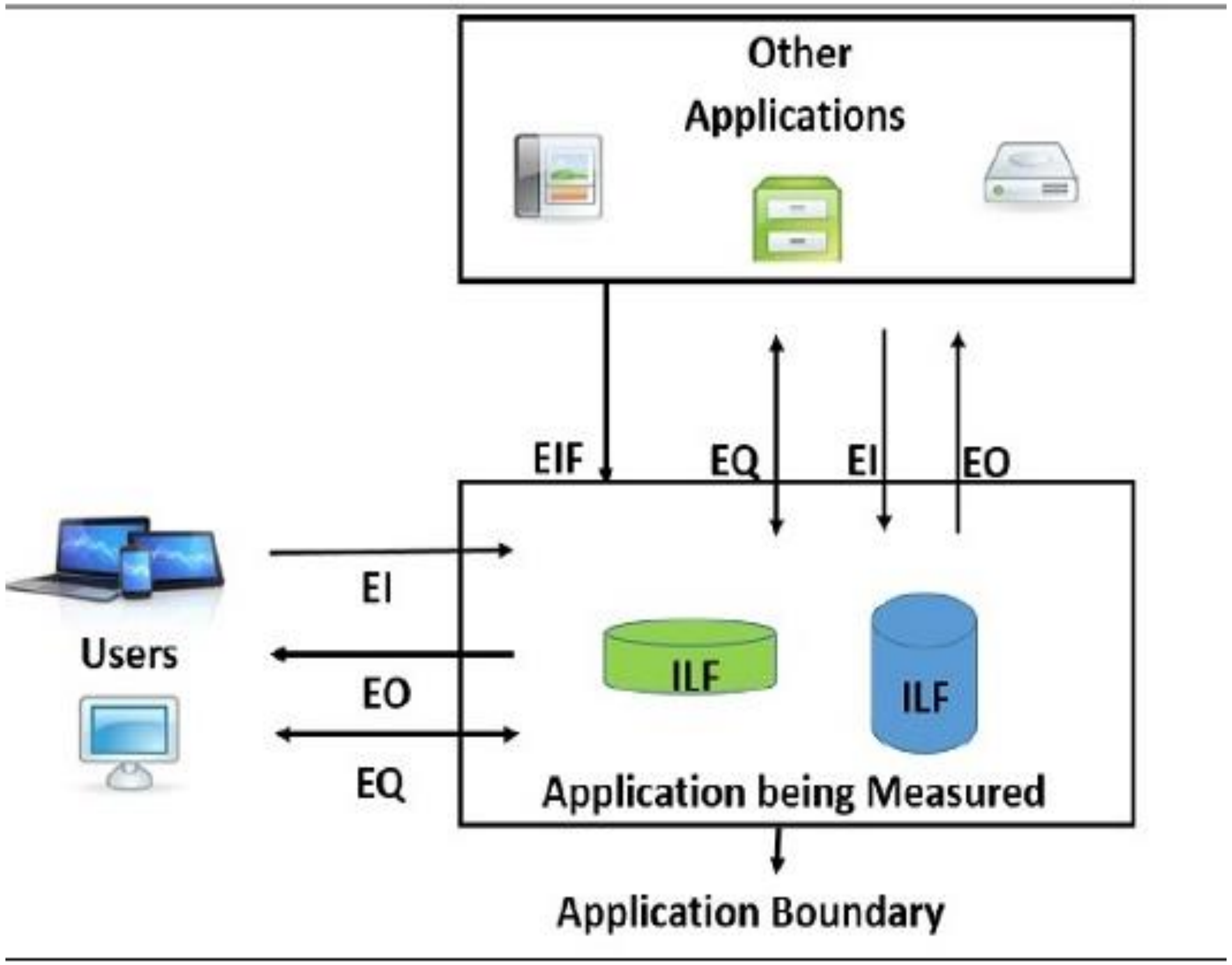
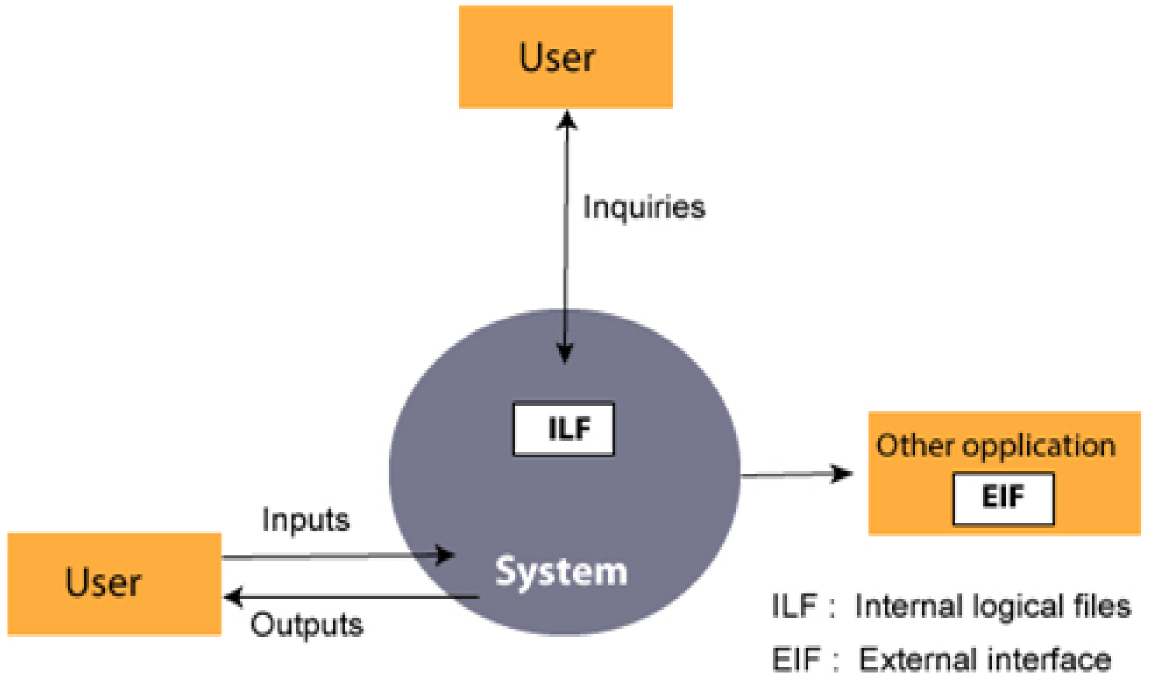
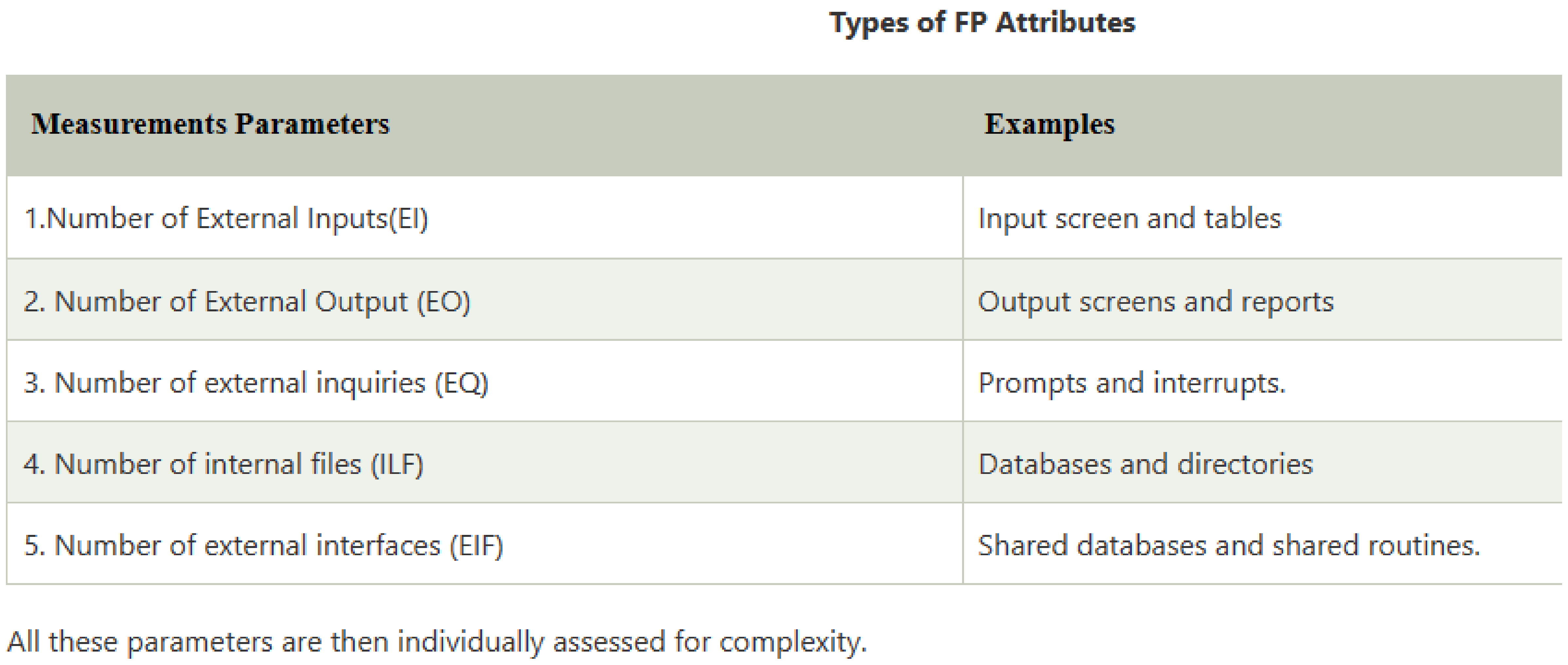


Figure 1: Application Boundary, Data Functions, Transaction Functions



FPAs Functional Units System

Function Point (FP)



Types of FP Attributes

Measurements Parameters	**Examples**
1.Number of External Inputs(EI)	Input screen and tables
2. Number of External Output (EO)	Output screens and reports
3. Number of external inquiries (EQ)	Prompts and interrupts.
4. Number of internal files (ILF)	Databases and directories
5. Number of external interfaces (EIF)	Shared databases and shared routines.

All these parameters are then individually assessed for complexity.

External inputs (EI)

- An external input (EI) is an elementary process that processes data or control information that comes from outside the application boundary
- The primary intent of an EI is to maintain one or more ILFs and/or to alter the behavior of the system
- Data may come from a data input screen or another application.
- An EI is how an application gets information
- Example:
  - Data entry by users
  - Data or file feeds by external applications

External outputs (EO)

- An external output (EO) is an elementary process that sends data or control information outside the application boundary
- The primary intent of an external output is to present information to a user through processing logic other than, or in addition to, the retrieval of data or control information
- The processing logic must contain at least one mathematical formula or calculation, create derived data, maintain one or more ILFs, or alter the behavior of the system
- Example:
  - Those items provided to the user that generate distinct application-oriented data (such as reports and messages, rather than the individual components of these)

External Inquires (EQ)

- An external inquiry (EQ) is an elementary process that sends data or control information outside the application boundary
- The primary intent of an external inquiry is to present information to a user through the retrieval of data or control information from an ILF or EIF
- The processing logic contains no mathematical formulas or calculations, and creates no derived data
- No ILF is maintained during the processing, nor is the behavior of the system altered
- Example:

  Interactive inputs requiring a response

Internal Logical Files (ILF)

- An ILF is a user-identifiable group of logically related data or control information maintained within the boundary of the application
- The primary intent of an ILF is to hold data maintained through one or more elementary processes of the application being counted
- Example:
  - Tables in a relational database
  - Files
  - Application control information, perhaps things like user preferences that are stored by the application

External Interface Files (EIF)

- An external interface file (EIF) is a user identifiable group of logically related data or control information referenced by the application, but maintained within the boundary of another application
- The primary intent of an EIF is to hold data referenced through one or more elementary processes within the boundary of the application counted
- This means an EIF counted for an application must be in an ILF in another application
- Example:
  - As for ILF, but maintained in a different system

Function-Oriented Metrics

Size Metric: **FUNCTION POINT ANALYSIS (FPA)**

Process used to calculate the function points

1. **Determine the type of project** for which the function point count is to be calculated. For example, development project (a new project) or enhancement project.
2. Identify the **counting scope** and the **application boundary.**
3. Identify **data functions** (internal logical functions and external interface files**) and their complexity.**
4. Identify **transactional functions** (external inputs, external outputs, and external queries) and their **complexity.**
5. Determine the **unadjusted function point count (UFP).**
6. Determine the **value adjustment factor (VAF) or (technical complexity factor) TCF ,** which is based on **14 general system characteristics (GSCs).**
7. Calculate the **adjusted function point count (AFP) or FP. (Same thing !!!)**

Unadjusted FP count (UFC)

- A complexity rating is associated with each count according to function point complexity weights, below:

$$\text{UFC} = \sum_{i=1}^{15} (\text{Number of items of variety } i) \times (\text{weight}_i)$$

  - Example: UFC = $4N_{EI} + 5N_{EO} + 4N_{EQ} + 7N_{EIF} + 10N_{ILF}$

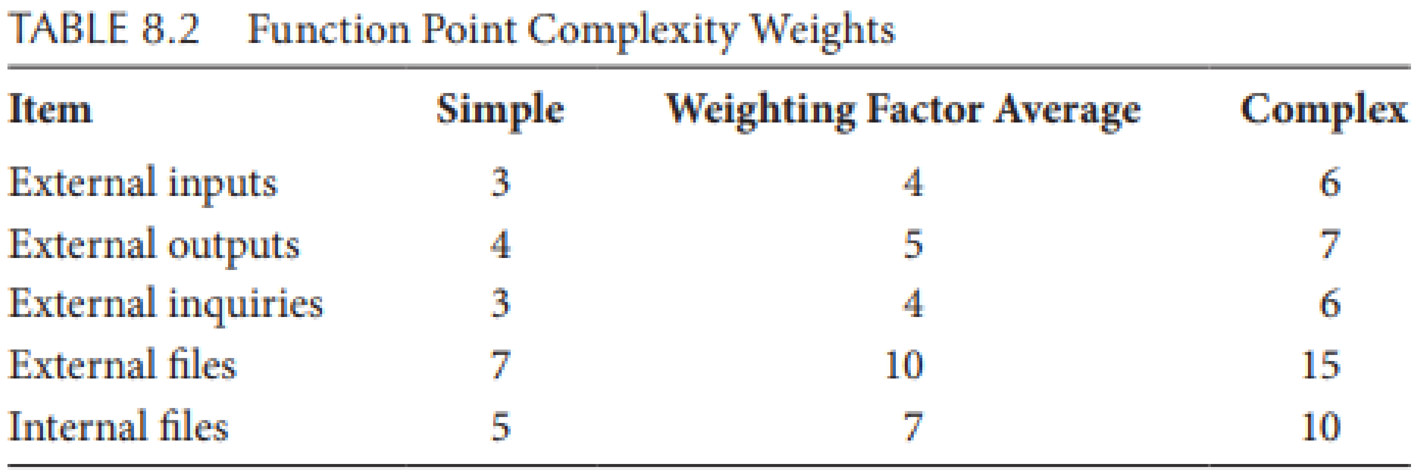


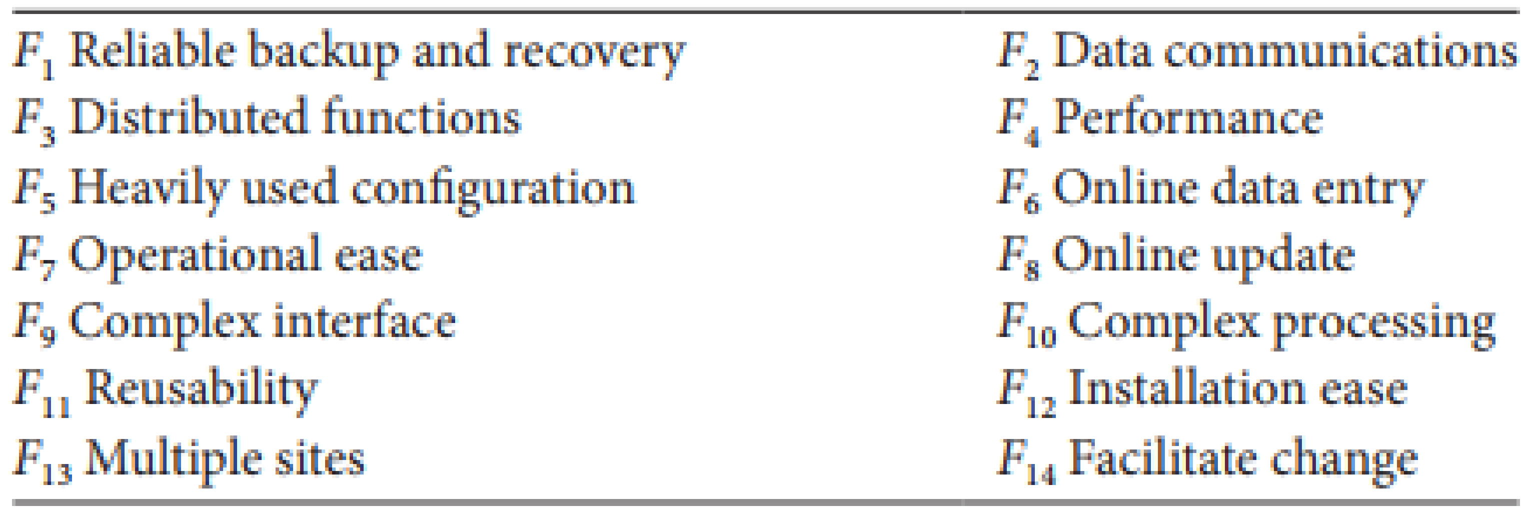
TABLE 8.2 Function Point Complexity Weights

Item	Simple	Weighting Factor Average	Complex
External inputs	3	4	6
External outputs	4	5	7
External inquiries	3	4	6
External files	7	10	15
Internal files	5	7	10

Technical Complexity Factors (TCF)

- Technical complexity factors (aka. Value Adjustment Factor (VAF)) is a weighted sum of 14 components, given below:

TABLE 8.3 Components of the Technical Complexity Factor



F_1 Reliable backup and recovery	F_2 Data communications
F_3 Distributed functions	F_4 Performance
F_5 Heavily used configuration	F_6 Online data entry
F_7 Operational ease	F_8 Online update
F_9 Complex interface	F_{10} Complex processing
F_{11} Reusability	F_{12} Installation ease
F_{13} Multiple sites	F_{14} Facilitate change

Technical Complexity Factors (TCF)

1. Reliable Backup and Recovery: The application need reliable backup and recovery
2. Data Communications: The data and control information used in the application are sent or received over communication facilities.
3. Distributed functions: Distributed data or processing functions are a characteristic of the application within the application boundary.
4. Performance: Application performance objectives, stated or approved by the user, in either response or throughput, influence the design, development, installation and support of the application.
5. Heavily Used Configuration: A heavily used operational configuration, requiring special design considerations, is a characteristic of the application.
6. On-line Data Entry: On-line data entry and control information functions are provided in the application.
7. Operational Ease: Operational ease is a characteristic of the application. Effective start-up, backup and recovery procedures were provided and tested during the system test phase.

Technical Complexity Factors (TCF)

8. On-line Update: The application provides on-line update for the internal logical files.
9. Complex interface: The application requires complex interface for input/output
10. Complex Processing: Complex processing is a characteristic of the application.
11. Reusability: The application and the code in the application have been specifically designed, developed and supported to be usable in other applications.
12. Installation Ease: Conversion and installation ease are characteristics of the application. A conversion and installation plan and/or conversion tools were provided and tested during the system test phase.
13. Multiple Sites: The application has been specifically designed, developed and supported to be installed at multiple sites for multiple organizations.
14. Facilitate Change: The application has been specifically designed, developed and supported to facilitate change.

Technical Complexity Factors (TCF)

- Each component is rated from 0 to 5, where 0 means the component is not relevant to the system and 5 means the component is essential.
- TCF can then be calculated as:

$$\mathrm{TCF} = 0.65 + 0.01\sum_{i=1}^{14} F_i$$

Min: 0
Max: 0.7

- VAF varies from 0.65 (if all Fj are set to 0) to 1.35 (if all Fj are set to 5)

FP = UFC × TCF

Counting Function Point (FP):

- **Step-1:**

```
F = 14 * scale
```

  Scale varies from 0 to 5 according to character of Complexity Adjustment Factor (CAF). Below table shows scale:

```
0 - No Influence
1 - Incidental
2 - Moderate
3 - Average
4 - Significant
5 - Essential
```

- **Step-2:** Calculate Complexity Adjustment Factor (CAF).

```
CAF = 0.65 + ( 0.01 * F )
```

- **Step-3:** Calculate Unadjusted Function Point (UFP).
  TABLE (Required)

AD

Function Units	Low	Avg	High
EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EIF	5	7	10

Multiply each individual function point to corresponding values in TABLE.

- **Step-4:** Calculate Function Point.

```
FP = UFP * CAF
```

Example 1:

Given the following values, compute function point when all complexity adjustment factor (CAF) and weighting factors are average.

User Input = 50; User Output = 40; User Inquiries = 35; User Files = 6; External Interface = 4

Explanation:

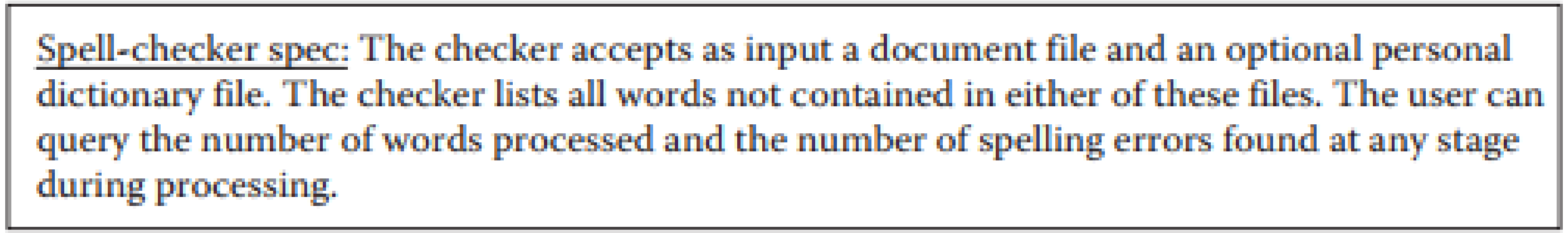
Step-1: As complexity adjustment factor is average (given in question), hence,
scale = 3.   F = 14 * 3 = 42

Step-2:
CAF = 0.65 + (0.01 * 42) = 1.07

Step-3: As weighting factors are also average (given in question) hence we will multiply each individual function point to corresponding values in TABLE.
UFP = (50*4) + (40*5) + (35*4) + (6*10) + (4*7) = 628

Step-4:
Function Point = 628 * 1.07 = 671.96
This is the required answer.

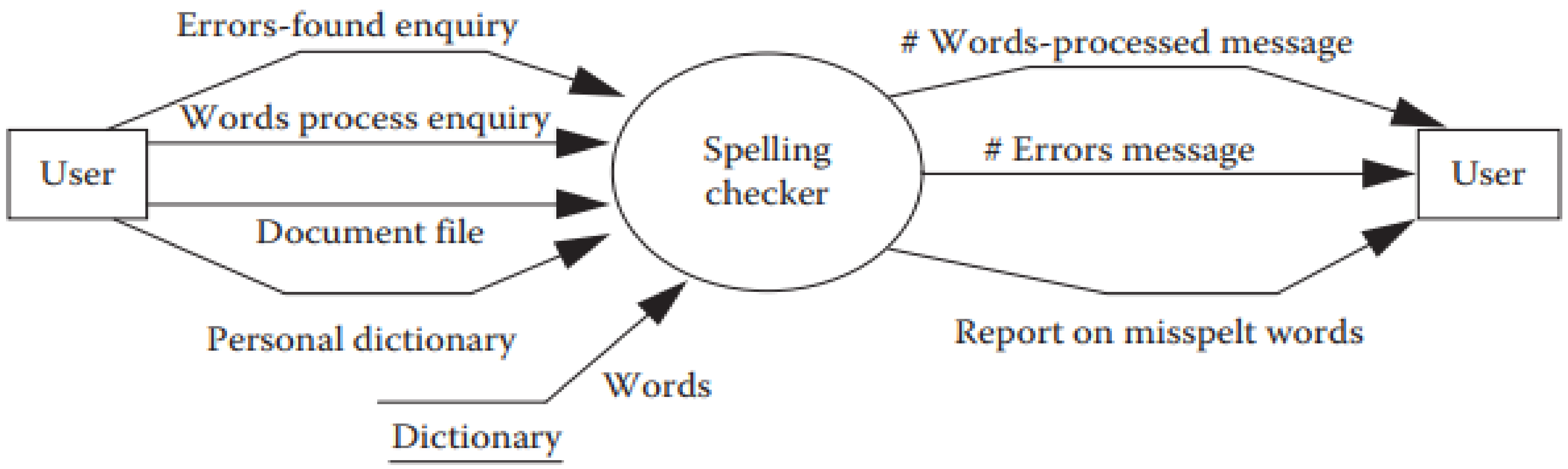
Example 2: Spell Checker


Spell-checker spec: The checker accepts as input a document file and an optional personal dictionary file. The checker lists all words not contained in either of these files. The user can query the number of words processed and the number of spelling errors found at any stage during processing.

- Checks all words in a document by comparing them to a list of words in the internal dictionary and an optional user-defined dictionary
- After processing the document sends a report on all misspelled words to standard output
- On request from user shows number of words processed on standard output
- On request from user shows number of spelling errors detected on standard output
- Requests can be issued at any point in time while processing the document file

Example: Spell Checker

- Spec to Diagram



A = # external inputs = 2, B = # external outputs = 3, C = # inquiries = 2,
D = # external files = 2, and E = # internal files = 1

Example: Spell Checker

$$UFC = 4A + 5B + 4C + 10D + 7E = 58$$

- Assume that:
  - *F*3, *F*5, *F*9, *F*11, *F*12, and *F*13 ------→ 0,
  - *F*1, *F*2, *F*6, *F*7, *F*8, and *F*14 ---------→ 3,
  - *F*4 and *F*10 ------→ 5

$$TCF = 0.65 + 0.01(18 + 10) = 0.93$$

- FP = UFC * TCF = 58 * 0.93 = 53.94

FP

- Suppose our historical database of project measurements reveals that it takes a developer an average of two person-days of effort to implement an FP.
- Then we may estimate the effort needed to complete the spelling checker as 118 days (i.e., 59 FPs multiplied by 2 days each).

FP Advantages

- Can be counted before design or code documents exist
- Can be used for estimating project cost, effort, schedule early in the project life-cycle
- Helps with contract negotiations
- Not dependent on languages

FP Limitations

- FP is a subjective measure: affected by the selection of weights by external users.
  - *Problems with subjectivity in the technology factor*.
  - *Problems with double counting.*
  - *Problems with subjective weighting.*
- Function point calculation requires a full software system specification. It is therefore difficult to use function points very early in the software development lifecycle.
- Problems with changing requirements.
- Not suitable for “complex” software, e.g., real-time and embedded applications.