Ch-23: PRODUCT METRICS

[up to 23.1.1 (pg:613-615), 23.2 up to 23.2.1 (pg:619-623)]

Product metrics

- Help software engineers ***gain insight*** into the design and construction of the software they build by focusing on specific, measurable attributes of software engineering work products.
- Software engineers use product metrics to ***help them build higher-quality software***.
    - Although product metrics for computer software are imperfect, they can ***provide a systematic way to assess quality*** based on a set of clearly defined rules.
    - They also ***provide an on-the-spot, rather than after-the-fact, insight***.
    - This enables software engineer to ***discover and correct potential problems before they become catastrophic defects***.

Importance of Product metrics

- There will always be a qualitative element to the creation of computer software.
- *Qualitative assessment may not be enough.*
- One needs an objective criteria to help guide the design of data, architecture, interfaces, and components.
- When testing - one needs quantitative guidance that will help in the selection of test cases and their targets.
- Product metrics provide a basis from which analysis, design, coding, and testing can be *conducted more objectively (tangibly) and assessed more quantitatively*.

Measurement process

Following are the Steps in the measurement process:

1. ***Formulation -*** *Derive the software measures and metrics that are appropriate* for the representation of software that is being considered.
2. ***Collection -*** *Data required to derive the formulated metrics are collected.*
3. ***Analysis*** - Once computed, *appropriate metrics are analyzed* based on preestablished guidelines and past data.
4. ***Interpretation*** - The *results of the analysis are interpreted* to *gain an understanding into the quality of the software*

5. ***Feedback -*** The results of the interpretation are transmitted to the software team.
   - ***It may lead to modification of requirements and design models, source code, or test cases***.
   - In some instances, results may also lead to ***modification of the software process itself***.

Work product - Product metrics that are computed from data collected from the requirements and design models, source code, and test cases.

23.1.1 Measures, Metrics, and Indicators

A *measure* provides a *quantitative indication* of the extent, amount, dimension, capacity, or size of some attribute of a product or process

When a single data point has been collected (e.g., the number of errors uncovered within a single software component), a measure has been established.

Measurement occurs as the result of the collection of one or more data points (e.g., a number of component reviews and unit tests are investigated to collect measures of the number of errors for each).

The IEEE glossary defines a *metric* as "*a quantitative measure of the degree to which a system, component, or process possesses a given attribute.*"

A software metric relates the individual measures in some way
(e.g., the average number of errors found per review or the average number of errors found per unit test).

A software engineer collects measures and develops metrics so that indicators will be obtained.

An *indicator* is *a metric or combination of metrics that provide insight* into the software process, a software project, or the product itself

An indicator provides insight that enables the project manager or software engineers to adjust the process, the project, or the product to make things better.

23.2 METRICS FOR THE REQUIREMENTS MODEL

- A software engineering project begins with creating a requirements model.
- The requirements are derived and a foundation for design is established.
- *Product metrics that provide insight into the quality of the analysis model* are necessary.
- Relatively few analysis and specification metrics have appeared in the literature
- It is possible to ***adapt metrics that are often used for project estimation*** and apply them in this context.
- These metrics examine the requirements model with the intention of *predicting the "size" of the resultant system*.
- ***Size is:***
    - *sometimes* (but not always) *an indicator of design complexity*
    - almost always an indicator of *increased coding, integration, and testing effort*.

23.2.1 Function-Based Metrics

[**Also refer:**
https://www.tutorialspoint.com/estimation_techniques/estimation_techniques_function_points.htm]

The function point (FP) metric can be used effectively as a means for measuring the functionality delivered by a system.

A function point (FP) is a unit of measurement to express the amount of business functionality, an information system (as a product) provides to a user.

FPs measure software size. They are widely accepted as an industry standard for functional sizing.

Using historical data, the **FP metric can be used to determine the following** in **the implemented system:**

(1) *estimate the cost or effort required* to design, code, and test the software

(2) *predict the number of errors* that will be encountered during testing

(3*) forecast the number of components and/or the number of projected source lines*

History of Function Point Analysis

The concept of Function Points was introduced by Alan Albrecht of IBM in 1979. The **International Function Point Users Group (IFPUG)** is a US-based non-profit worldwide organization of Function Point Analysis metric software users. The member-governed organization, was founded in 1986. IFPUG owns Function Point Analysis (FPA) as defined in ISO standard 20296:2009 which specifies the definitions, rules and steps for applying the IFPUG's functional size measurement (FSM) method.

Function Points (FP) Counting is governed by a standard set of rules, processes and guidelines as defined by the IFPUG. These are published in Counting Practices Manual (CPM).

Function Point Analysis (FPA) technique *quantifies the functions contained within software* in terms that are meaningful to the software users. *FPs consider the number of functions being developed based on the requirements specification.*

There are two types of functions:

- ***Data Functions*** - made up of internal and external resources that affect the system.
  - **Internal Logical Files (ILFs)**
  - **External Interface Files (EIFs)**
- ***Transaction Functions*** - Transaction functions are made up of the processes that are exchanged between the user, the external applications and the application being measured.
  - **External Inputs (EIs)**
  - **External Outputs (EOs)**
  - **External Inquiries (EQs)**

Function points are derived using an empirical relationship based on countable (direct) measures of software's information domain and qualitative assessments of software complexity.

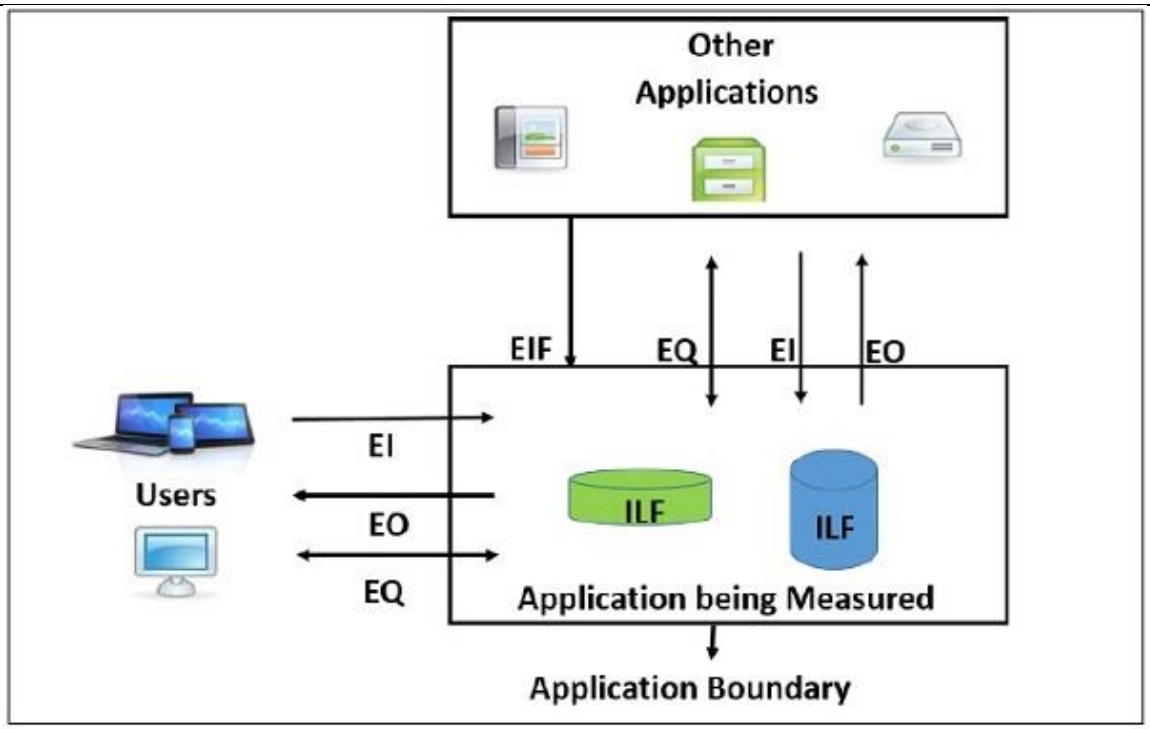

Figure: Application Boundary, Data Function, Transition Function

Image Source:
https://www.tutorialspoint.com/estimation_techniques/estimation_techniques_function_points.htm

Information domain values are defined in the following manner:

Number of external inputs (EIs): Each external input originates from a user or is transmitted from another application and provides distinct application-oriented data or control information. Inputs are often used to update internal logical files (ILFs). Inputs should be distinguished from inquiries, which are counted separately.

Number of external outputs (EOs): Each external output is derived data within the application that provides information to the user.

In this context external output refers to *reports*, *screens*, *error messages*, etc. Individual data items within a report are not counted separately.

Number of external inquiries (EQs): An external inquiry is defined as an online input that results in the generation of some immediate software response in the form of an online output (often retrieved from an ILF).

Number of internal logical files (ILFs): Each internal logical file is a logical grouping of data that resides within the application's boundary and is maintained via external inputs.

Number of external interface files (EIFs): Each external interface file is a logical grouping of data that resides external to the application but provides information that may be of use to the application.

FP Counting Process involves the following steps:

Step 1: Determine (measure) the information domain values (data function/ transition functions) i.e. EIs, EOs, EQs, ILFs & EIFs

Step 2: Determine the functional complexity of these values

Step 3: Calculate functional size (Unadjusted Function Point Count (UFP))

Step 4: Determine Value Adjustment Factor (VAF)

Step 5: Calculate Adjusted Function Point count (AFP)

Example 1 (*SafeHome* software):

Let us take an example of the *SafeHome* software (Please refer detail from the book, pg 620-623)

Step 1: Determine (measure) the information domain values (data function/ transition functions) i.e. EIs, EOs, EQs, ILFs & EIFs

The information domain values were counted as below:

Information domain values (Measurement parameters)	**Count**
Number of User Inputs, EIs	3
Number of User Outputs, EOs	2
Number of User Inquires, EQs	2
Number of Files, ILFs	1
Number of External Interfaces, EIFs	4

Step 2: Determine the functional complexity of the above values

A complexity value is associated with each count.
Organizations that use function point methods develop criteria for determining whether a particular entry is **simple, average, or complex (low, average or high)**.
Weighing factor is also defined for each (standard defined in CPM). (see table below).
(Note: The determination of complexity is somewhat subjective.)

Information domain values	Weighing Factors		
	Simple	Average	Complex
Number of User Inputs, EIs	3	4	6
Number of User Outputs, EOs	4	5	7
Number of User Inquires, EQs	3	4	6
Number of Files, ILFs	7	10	15
Number of External Interfaces, EIFs	5	7	10

Step 3: Calculate Functional Size (Unadjusted Function Point Count)

In this example, assuming a **simple** analysis model representation

Information domain values	Count	Weighing factor	FP Count = Count x Weighing factor
		Simple	
Number of User Inputs, EIs	3	3	3x3=9
Number of User Outputs, EOs	2	4	2x4=8
Number of User Inquires, EQs	2	3	2x3=6
Number of Files, ILFs	1	7	1x7=7
Number of External Interfaces, EIFs	4	5	4x5=20
Count Total or **Unadjusted Function Point Count (UFP)**			9+8+6+7+20 **= 50**

Thus, Count Total (Unadjusted Function Point Count (UFP)) = 50

Step 4: Determine the Value Adjustment Factor

- The Value Adjustment Factor (VAF) is based on 14 **General System Characteristic (**GSCs) that rate the general functionality of the application being counted.
- GSCs are user business constraints independent of technology.
- Each characteristic has associated descriptions to determine the degree of influence. The **degree of influence** range is on a scale of **zero (no influence, i.e. not important or applicable) to five (strong influence, i.e. absolutely essential)**

Rating	Degree of Influence
0	Not present, or no influence i.e. not important or applicable
1	Incidental influence
2	Moderate influence
3	Average influence
4	Significant influence
5	Strong influence throughout absolutely essential

The F_i (i= 1 to 14) are *value adjustment factors* (VAF) based on responses to the following questions (General System Characteristic (GSCs)):
[Each of these questions is answered using a scale that ranges from to 5.]
1. Does the system require reliable backup and recovery?
2. Are specialized data communications required to transfer information to or from the application?
3. Are there distributed processing functions?
4. Is performance critical?
5. Will the system run in an existing, heavily utilized operational environment?
6. Does the system require online data entry?
7. Does the online data entry require the input transaction to be built over multiple
screens or operations?
8. Are the ILFs updated online?

9. Are the inputs, outputs, files, or inquiries complex?
10. Is the internal processing complex?
11. Is the code designed to be reusable?
12. Are conversion and installation included in the design?
13. Is the system designed for multiple installations in different organizations?
14. Is the application designed to facilitate change and ease of use by the user?

After determining the degree of influence for each of the 14 GSCs, the sum of the values of the 14 GSCs is calculated. This is termed as Total Degree of Influence (TDI).

Total Degree of Influence (TDI) = $\sum F_i = \sum^{14}$ Degrees of Influence

For the purposes of this example, we assume that $\sum$ (Fi) is 46 (a moderately complex product). Therefore,

i.e. TDI = $\sum F_i$ = 46

Next, calculate the **Value Adjustment Factor (VAF) or Complexity adjustment factor**

Complexity adjustment factor

= Value Adjustment Factor (VAF) = 0.65 + (0.01 x TDI)

= 0.65 + (0.01 x 46)

= 0.65 + 0.46

= **1.11**

[Note: The constant values in this equation (0.65) and the weighting factors that are applied to information domain counts are determined empirically.**]**

Step 5: Calculate Adjusted Function Point Count

The unadjusted FP count is the functional size that we have calculated in Step 3.

Adjusted FP Count = Unadjusted FP Count × VAF

= 50 x 1.11 = 55.5

= 56 (approx.)

****

SUMMARIZING:

FP Count = Count of an Information domain value (Measurement parameter)

x Weighing factor for that parameter

Count Total = Unadjusted Function Point Count (UFP)

= Total of all five FP count (for the five information domain values)

Total Degree of Influence (TDI) = $\sum F_i$ = $\sum^{14}$ Degrees of Influence

Complexity adjustment factor = Value Adjustment Factor (VAF) = 0.65 + (0.01 x TDI)

Adjusted FP Count = Unadjusted FP Count × VAF

NOTE:

- Each GSC can vary from 0 to 5
- TDI can vary from (0 × 14) to (5 × 14), i.e. 0 (when all GSCs are low) to 70 (when all GSCs are high) i.e. 0 ≤ TDI ≤ 70.
- Hence, VAF can vary in the range from 0.65 (when all GSCs are low) to 1.35 (when all GSCs are high), i.e., 0.65 ≤ VAF ≤ 1.35.
- As the VAF can vary from 0.65 to 1.35, the VAF exerts an influence of ±35% on the final adjusted FP count.

Based on the projected FP value derived from the requirements model, the ***project team can estimate the overall implemented size*** of the *SafeHome* user interaction function. [Details will be discussed in Chapter 25]

- Assume that past data indicates that one FP translates into 60 lines of code (an object-oriented language is to be used)
- 12 FPs are produced for each person-month of effort.

(Application 1 of FPs) These historical data ***provide the project manager with important planning information that is based on the requirements model rather than preliminary estimates.***

Assume further that past projects have found:

- an average of three errors per function point during requirements and design reviews
- and four errors per function point during unit and integration testing.

(Application 2 of FPs) These data can ultimately ***help the team assess the completeness of their review and testing activities.***

[Note*: Function points can also be computed from UML class and sequence diagrams]*

Example 2:

Compute the Function Point value for a project with the following domain characteristics:

Measurement parameters	Count	Weighing Factors		
		Low	Average	High
Number of User Inputs	12	3	4	6
Number of User Outputs	21	4	5	7
Number of User Inquires	6	3	4	6
Number of Files	6	7	10	15
Number of External Interfaces	3	5	7	10

Assume the measurement parameters have average complexity. Also assume that the complexity adjustment value is 1.15.

Answer:

Given that the measurement parameters have average complexity.

Computing the Count Total, i.e. Unadjusted Function Point Count (UFP)

Measurement parameters	Count	Weighing factor	FP Count = Count x Weighing factor
		Average	
Number of User Inputs	12	4	12x4=48
Number of User Outputs	21	5	21x5=105
Number of User Inquires	6	4	6x4=24
Number of Files	6	10	6x10=60
Number of External Interfaces	3	7	3x7=21
Count Total or **Unadjusted Function Point Count (UFP)**			48+105+24+60+21 **= 258**

Thus **UFP = 258**

Given, **Complexity adjustment factor = Value Adjustment Factor (VAF) = 1.15**

Adjusted FP Count = Unadjusted FP Count × VAF

= 258 x 1.15 = 296.7

Hence, Adjusted FP Count = 297 (approx.)

****

Example 3:

A system has 2 external inputs, 5 external outputs, 3 external queries, manages 5 internal logical files, and interfaces with 3 external legacy systems. All the data are of simple complexity 3, 4, 3, 7, and 5 respectively. The overall system is relatively simple. Compute Functional Point for the system.

Answer:

The given values of EIs, EOs, EQs, ILFs & EIFs are 2, 5, 3, 5, and 3 respectively.

Given that all the data are of simple complexity 3, 4, 5, 7, and 5 respectively.

Computing the Count Total, i.e. Unadjusted Function Point Count (UFP)

Measurement parameters	**Count**	**Weighing factor**	**FP Count = Count x Weighing factor**
		Simple	
Number of User Inputs	2	3	2x3=6
Number of User Outputs	5	4	5x4=20
Number of User Inquires	3	3	3x3=9
Number of Files	5	7	5x7=35
Number of External Interfaces	3	5	3x5=15
Count Total or **Unadjusted Function Point Count (UFP)**			6+20+9+35+15 **= 85**

Thus **UFP = 85**

Given that the overall system is relatively simple. The weighing factor can be assumed as 1 or 2. *Hence two different answers are possible*

Assume weighing factor for each of the 14 questions is 1

Total Degree of Influence (TDI) = $\sum F_i$ = 14x1 **=14**

Complexity adjustment factor = Value Adjustment Factor (VAF)

= 0.65 + (0.01 x TDI) = 0.65 + (0.01 x 14) = 0.65 + 0.14

= 0.79

Hence, Adjusted FP Count = Unadjusted FP Count × VAF = 85 x 0.79 = 67.15

i.e Adjusted FP Count = 68 (approx.)

Assume weighing factor for each of the 14 questions is 2

Total Degree of Influence (TDI) = $\sum F_i$ = 14x2 **= 28**

Complexity adjustment factor = Value Adjustment Factor (VAF)

= 0.65 + (0.01 x TDI) = 0.65 + (0.01 x 28) = 0.65 + 0.28

= 0.93

Hence, Adjusted FP Count = Unadjusted FP Count × VAF = 85 x 0.93 = 79.05

i.e <u>Adjusted FP Count = 80 (approx.)</u>

****

QUESTIONS

Question 1. Compute the Function Point value for a project with the following domain characteristics:

Measurement parameters	Count	Weighing Factors		
		Low	Average	High
Number of User Inputs	9	3	4	6
Number of User Outputs	15	4	5	7
Number of User Inquires	6	3	4	6
Number of Files	6	7	10	15
Number of External Interfaces	3	5	7	10

Assume the measurement parameters equally divided among low, average and high complexity. Also assume that the complexity adjustment value is 1.05.

Answer 1: UFP = 231; VAF = 1.05; AFP = 243

Question 2. A system has 5 external inputs, 8 external outputs, 3 external queries, manages 5 internal logical files, and interfaces with 3 external legacy systems (3 EIFs). All the data are of high complexity 6, 7, 6, 15, and 10 respectively. Also assume that the complexity adjustment value is 1.07. Compute Functional Point for the system.

Answer 2: UFP = 209; VAF = 1.07; AFP = 228