SWE 4603

Software Testing and Quality Assurance

Lecture 10

Prepared By Maliha Noushin Raida, Lecturer, CSE
Islamic University of Technology

Lesson Outcome

- Entities to be measured for the software: process, product, and resource
- Recognition of attributes before measurement, as they are important for designing software metrics
- Line-of-code metrics
- Halstead metrics
- Function point analysis metrics

Week on:

- Chapter 10: Software Metrics

Software Metrics

Measure

- provides a quantitative indication of the extent, amount, dimension, capacity or size of some attribute of a process or product.

  Example: Number of defects found in component testing. LOC of each component.

Measurement

- The act of collecting a measure.
- Example: Collecting the defect counts. Counting LOC

Software Metrics

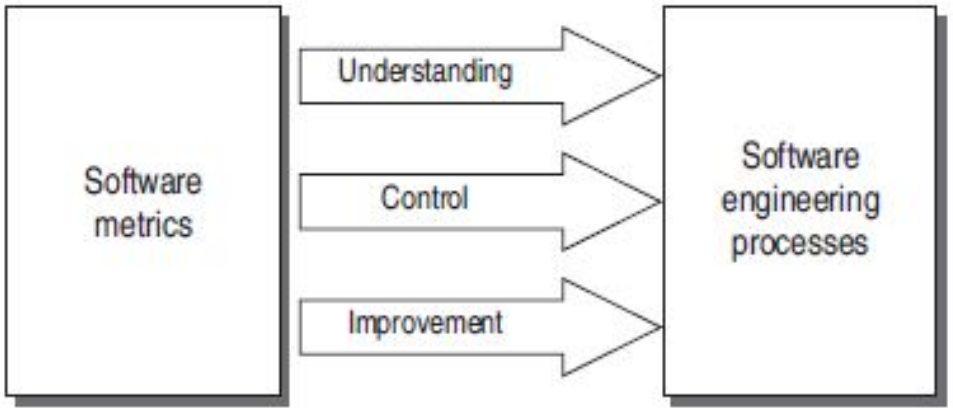
SW Metrics refers to a range of measurements for computer software that enable software people to gain insight into the project:

- To improve the Process and the Product
- Assist in Estimation
- Productivity Assessment
- Quality Control
- Project Control

Software Metrics

Need of SW Measurement:

- Measurements are a key element for controlling software engineering processes.
- software measurement is needed for the following activities:
  - Understanding
  - Control
  - Improvement



Software Metrics

SW Metrics Definition:

The IEEE Standard Glossary of Software Engineering Terms defines a metric as **‘a quantitative measure of the degree to which a system component or process possesses a given attribute.’**

Software Metrics

Classification of SW Metrics

Product vs Process Metrics

- **Product metrics** are measures of the software product at any stage of its development, from requirements to installed system.
- **Product metrics** may measure the complexity of the software design, the size of the final program, or the number of pages of documentation produced.
- **Process metrics**, on the other hand, are measures of the software development process, such as the overall development time, type of methodology used, or the average level of experience of the programming staff.

Software Metrics

Classification of SW Metrics

Primitive vs computed Metrics

- Primitive metrics are those metrics that can be directly observed, such as the program size in LOC, the number of defects observed in unit testing, or the total development time for the project.
- Computed metrics are those that cannot be directly observed but are computed in some way from other metrics. For example, productivity metrics like LOC produced per person-month or product quality like defects per thousand lines of code.

Software Metrics

Classification of SW Metrics

Private vs Public Metrics

- This classification is based on the use of different types to process data.
- Examples of private metrics include defect rates (by individual and by module) and errors found during development.
- Public metrics integrate information that originally was private to individuals and teams. Project-level defect rates, effort, calendar times, and related data are collected and evaluated in an attempt to uncover indicators that can improve organizational process performance.

Software Metrics

Entities to be measured:

The entities considered in software measurement are:

Processes Any activity related to software development.

Product Any artifact produced during software development.

Resource People, hardware, or software needed for a process.

The attributes of an entity can be **internal** or **external**.

- SW size is an internal attribute of any software measurement.
- Productivity, an external attribute of a person, clearly depends on many factors such as the kind of process and the quality of the software delivered.

Software Metrics

Size Metric:

- Software size is an important metric to be used for various purposes.
- Difficult to Measure
- Various approaches are used to measure SW size
  - LINE OF CODE (LOC)
  - TOKEN COUNT (HALSTEAD PRODUCT METRICS)
  - FUNCTION POINT ANALYSIS (FPA)

Software Metrics

LINE OF CODE (LOC)

- This metric is based on the number of lines of code present in the program.
- The lines of code are counted to measure the size of a program.
- The comments and blank lines are ignored during this measurement.
- The LOC metric is often presented on thousands of lines of code (KLOC).

Software Metrics

TOKEN COUNT (HALSTEAD SOFTWARE METRICS)

- LOC is not consistent, because all lines of code are not at the same level.
- Some lines are more difficult to code than others.
- **Halstead** stated that any software program could be measured by counting the number of operators and operands.
- From these set of operators and operands, he defined a number of formulae to calculate the **vocabulary**, the **length**, and the **volume** of the software program.

Software Metrics

TOKEN COUNT (HALSTEAD SOFTWARE METRICS)

- **Halstead Metric**

  Program Vocabulary

  - It is the number of unique operators plus the number of unique operands as given below:

    $$n = n1 + n2$$

    where n = program vocabulary

    n1 = number of unique operators

    n2 = number of unique operands

Software Metrics

TOKEN COUNT (HALSTEAD SOFTWARE METRICS)

- **Halstead Metric**

  Program Length

  - It is the total usage of all the operators and operands appearing in the implementation.

    $$N = N1 + N 2$$

    where N = program length

    N1 = all operators appearing in the implementation

    N2 = all operands appearing in the implementation

Software Metrics

TOKEN COUNT (HALSTEAD SOFTWARE METRICS)

- **Few Halstead Metric**

  Program Volume

  - The volume refers to the size of the program and it is defined as the program length times the logarithmic base 2 of the program vocabulary.

    $$V = N \log_2 n$$

    where V = program volume

    N = program length

    n = program vocabulary

Software Metrics

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 early in the development process.
- The size of an application is measured in **function points**.
- FP can be used for:
  - Estimate the effort or cost of code and test the software system
  - Forecast the number of components and/or the number of source lines in the projected system.

Software Metrics

FUNCTION POINT ANALYSIS (FPA)

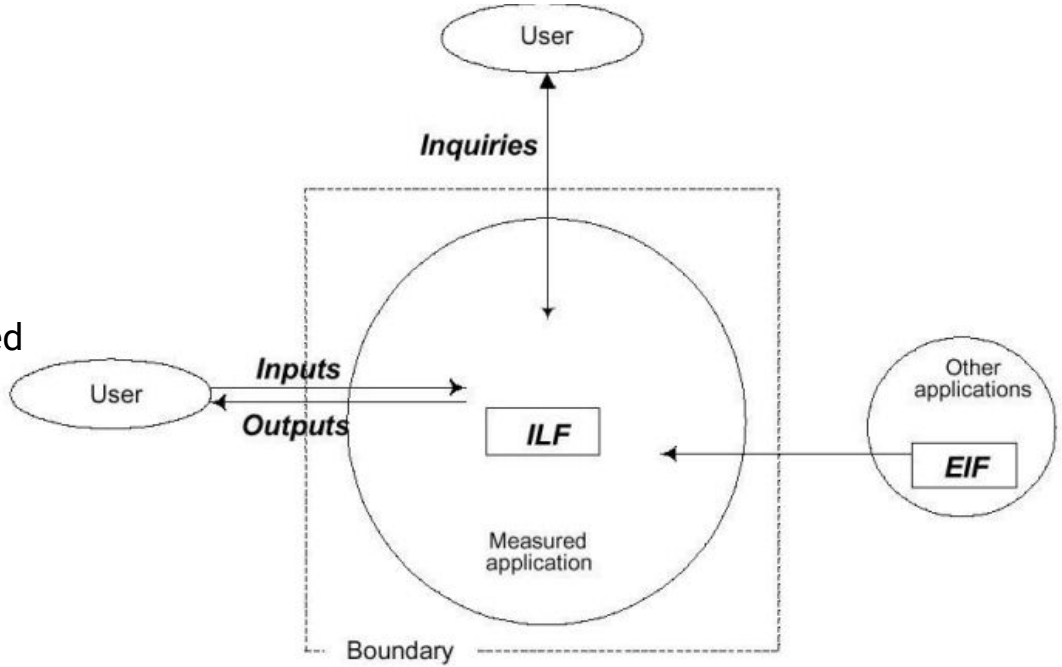
Process used to calculate the function points

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)**, which is based on **14 general system characteristics (GSCs).**
7. Calculate the **adjusted function point count (AFP).**

Software Metrics

FUNCTION POINT ANALYSIS (FPA)

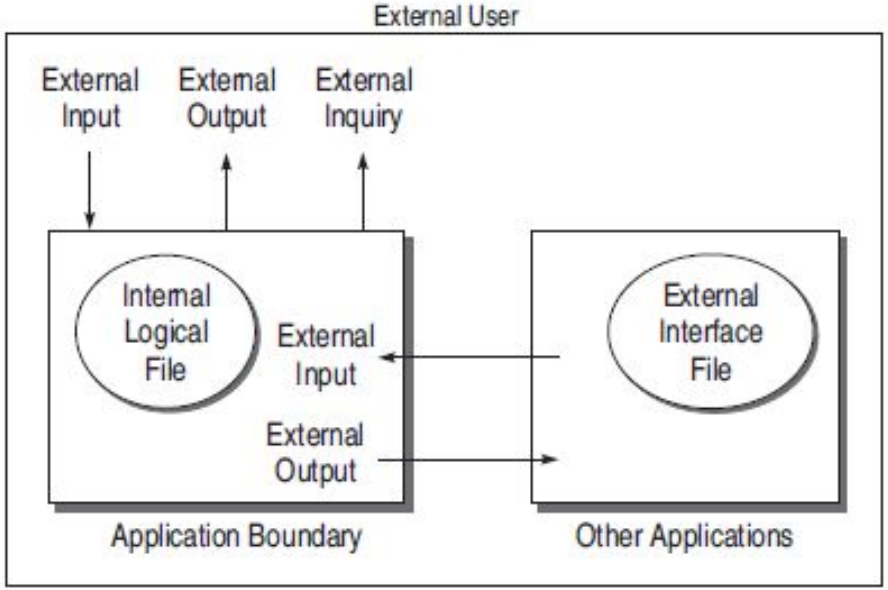
- The first step in calculating FP is to identify the counting boundary.
- Counting boundary: The border between the application or project being measured and external applications or the user domain.
- A boundary establishes which functions are included in the function point count



Software Metrics

FUNCTION POINT ANALYSIS (FPA)

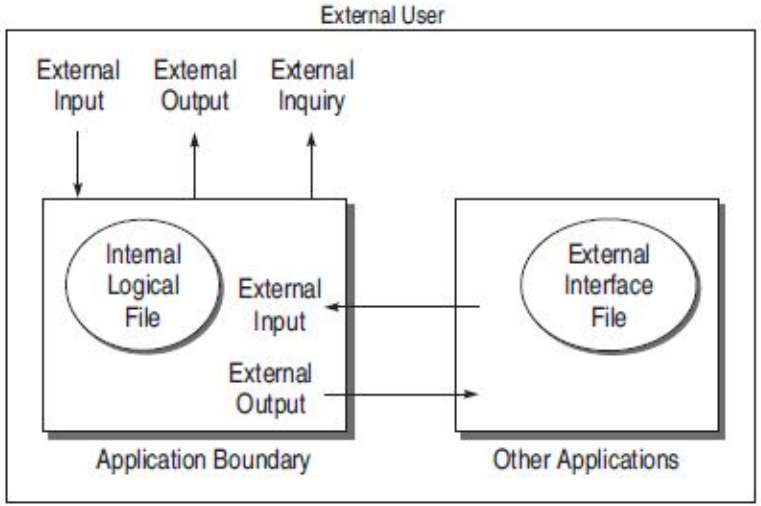
- **Two data function types:**
  - **Internal Logical Files (ILF):** these file are the master or transaction files that the system interacts with during its session.
  - **External Interface Files(EIF):** represent the data that your application will use/reference, but data that is not maintained by your application.



Software Metrics

FUNCTION POINT ANALYSIS (FPA)

- **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.



Software Metrics

FUNCTION POINT ANALYSIS (FPA)

- **Calculating UFP:**
  - Count all five components, i.e. ILF, EIF, EI, EO, and EQ of an application and determine the level of the component as low, average, or high, using a set of prescriptive standards.
  - Nevertheless, the determination of complexity is somewhat subjective.
  - Count the number of all five components. The sum of each count is multiplied by an appropriate weight using Table below.

Components	Function Levels		
	Low	Average	High
ILF	X7	X10	X15
EIF	X5	X7	X10
EI	X3	X4	X6
EO	X4	X5	X7
EQ	X3	X4	X6

  - Add all the five results calculated in the previous step. This is the final UFP.

Software Metrics

- Calculating Adjusted Function Point:
  - A **value adjustment factor (VAF)** is used as a multiplier of the unadjusted function point (UFP) count in order to calculate the adjusted function point (AFP) count of an application.
  - To calculate the VAF, we evaluate the 14 GSCs(General system Characteristics) on a scale of 0–5 to determine the degree of influence(DI) for each GSC description.
  - Add the DIs for all 14 GSCs to produce the total degree of influence (TDI).
  - Use the TDI in the following equation to compute VAF.

    VAF = (TDI × 0.01) + 0.65
  - The final adjusted function point is calculated as,

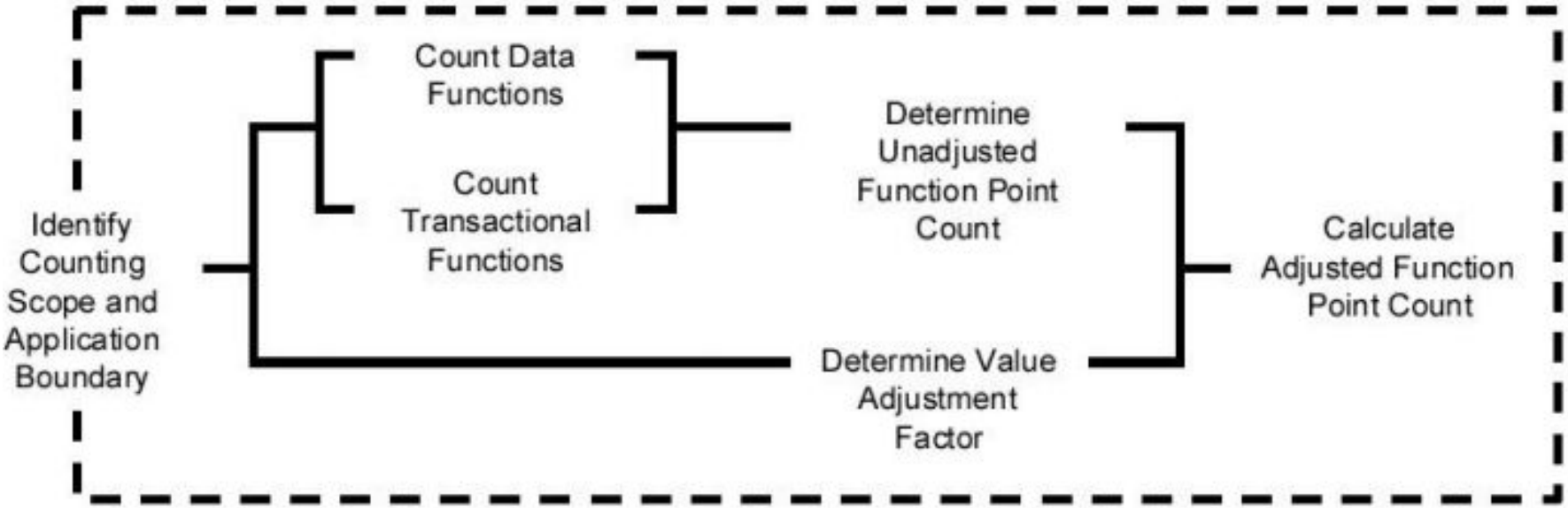
    AFP = UFP × VAF

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

Factor	Meaning
F1	Data communications
F2	Performance
F3	Transaction rate
F4	End user efficiency
F5	Complex processing
F6	Installation ease
F7	Multiple sites
F8	Distributed data processing
F9	Heavily used configuration
F10	Online data entry
F11	Online update
F12	Reusability
F13	Operational ease
F14	Facilitate change

Software Metrics

FUNCTION POINT ANALYSIS (FPA)



Software Metrics

FUNCTION POINT ANALYSIS (FPA)

Example:

Consider a project with the following parameters: EI = 50, EO = 40, EQ = 35, ILF = 06, and ELF = 04. Assume all weighing factors are **average**. In addition, the system requires **Significant** performance, **average** end-user efficiency, **moderate** distributed data processing, and **Significant** data communication. Other GSCs are **incidental**. Compute the function points using FPA.

Measurement Parameter	Count		Weighing factor			
			Simple	Average	Complex	
1. Number of external inputs (EI)	50	*	3	4	6 =	200
2. Number of external Output (EO)	40	*	4	5	7 =	200
3. Number of external Inquiries (EQ)	35	*	3	4	6 =	140
4. Number of internal Files (ILF)	06	*	7	10	15 =	60
5. Number of external interfaces(EIF)	04	*	5	7	10 =	28
UFP Count-total →						628

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

General System Characteristics (GSCs)	Degree of Influence (DI) 0 - 5
1. Data Communications	4
2. Distributed Data Processing	2
3. Performance	4
4. Heavily Used Configuration	1
5. Transaction Rate	1
6. Online Data Entry	1
7. End-User Efficiency	3
8. Online Update	1
9. Complex Processing	1
10. Reusability	1
11. Installation Ease	1
12. Operational Ease	1
13. Multiple Sites	1
14. Facilitate Change	1
Total Degree of Influence (TDI)	23
Value Adjustment Factor (VAF)	(23*0.01)+0.65 =0.88
	VAF = (TDI * 0.01) + 0.65

As we know,

Function Point = VAF* UFP
= 0.88*628
= **552.64**

Function Points Vs LOC

Function Point	Line of Code
Function Point metric is specification-based.	LOC metric is based on analogy.
Function Point metric is language independent.	LOC metric is dependent on language.
Function Point metric is user-oriented.	LOC metric is design-oriented.
Function Point metric is extendible to Line of Code.	It is changeable to FP (i.e, backfiring)
Function Point is used for data processing systems	LOC is used for calculating the size of the computer program
Function Point can be used to portray the project time	LOC is used for calculating and comparing the productivity of programmers.