Chapter 3
Maintenance Measurements

- Maintenance Metrics
- Maintenance Cost Estimation

" ...if you can measure what you are speaking about and express it in numbers you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge of it is of a meagre and unsatisfactory kind"

Lord Kelvin

Prepared By Mintesinot A.

Maintenance Measurements

- ***software measurement*** is the process of objectively and empirically quantifying an attribute of a software system and the process connected with its development, use, maintenance and evolution.
- The above definition applies both to development of new system and maintenance of existing system.
- In general, there are three software maintenance-related entities whose attributes can be subjected to measurement: process, product and resource.
    - A **process** is any software-related activity such as change analysis, specification, design, coding and testing.
    - A **resource** is input to a process, for example personnel, hardware and software.
    - A **product** is any intermediate and final output resulting from a software process such as system documentation, program listings, test data, source code and object code.

Maintenance Measurements

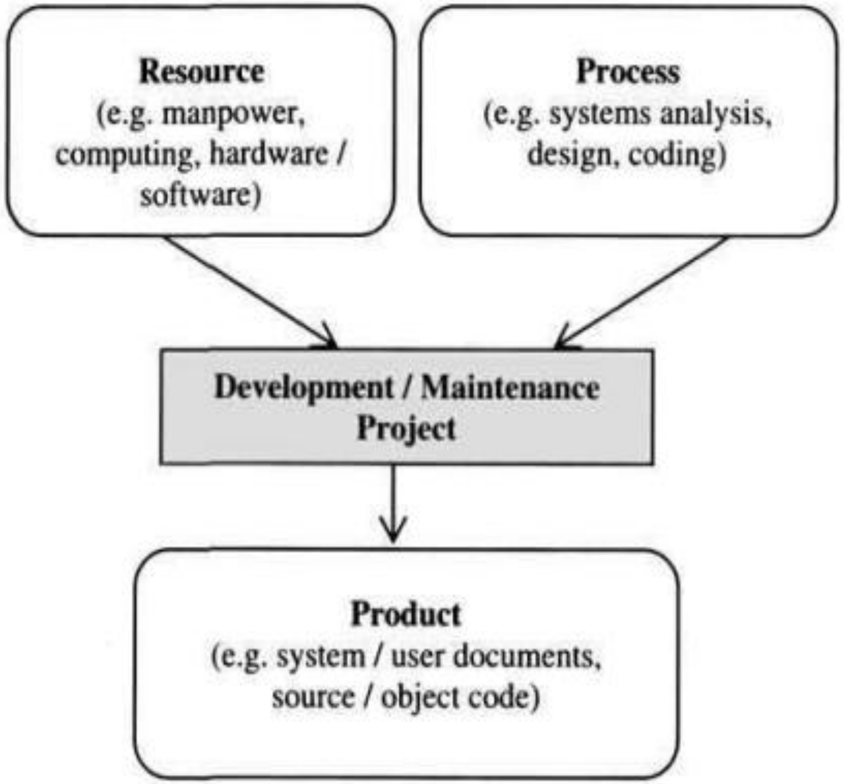


Figure: Relation between a resource, process and product

Maintenance Measurements

- In software measurement, two types of attribute can be identified: internal and external.
- An **internal attribute** is one which can be measured in terms of the process, product or resource itself.
  - ✓ For example, complexity, modularity and reusability are internal attributes of the source code of a program.
- An **external attribute** is one which can only be measured with respect to the relation of a process, product or resource to its environment,
  - ✓ for example the maintainability of program source code or productivity of software personnel.

The Importance of Integrity in Measurement

- A measurement procedure must demonstrate a number of characteristics. It must be

i. **Empirical:** The result of measurement should describe empirically established facts.

- Finkelstein captured the importance of this when he said that the precise, concise, objective and empirical nature of measurement 'gives its primacy in science'.

ii. **Objective:** During measurement, observations should be carried out with integrity, objectively, reliably, efficiently and without bias or ambiguity.

iii. **Encodable:** An attribute can be encoded or characterized using different symbols such as numbers and graphic representations.

Maintenance Metrics

Metric- A criterion to determine the difference or distance between two entities, like the distance of a query and a document in Information Retrieval Systems.

- A well-known metric is the metric of Euclid, which measures the shortest distance between two points"

Objectives Of Software Measurement

- **Evaluation*:*** To evaluate different methods, program libraries and tools before arriving at a decision as to which is best suited to a given task.
- **Control*:*** To control the process of software change to ensure that change requests are dealt with promptly and within budget.
  - As DeMarco says, "you cannot control what you cannot measure"
- **Assessment*:*** In order to control a process or product, it is important to be able to assess or to characterize it first.
  - A manager may need to assess a system to determine whether or not it is economically feasible to continue maintaining it.
  - Also, in order to determine whether or not the maintenance process being used is achieving or will achieve the desired effect, an assessment of the process must be undertaken.

Objectives Of Software Measurement…

- ***Improvement:*** To improve various characteristics of the software system or process such as quality and productivity.

- ***Prediction:*** To make predictions about various aspects of the software product, process & cost.
  - ✓ For instance, measures obtained from program code can be used to predict the time required to implement a given change.
  - ✓ These measures can assist a manager in the allocation of time, personnel, hardware and software resources to a maintenance project.

Maintenance Measures

- There are several measures that maintainers may need in order do their job.

➢ In theory these measures can be derived from the attributes of the software system, the maintenance process and personnel.

➢ In practice, the most commonly used source of measures is the software system, specifically the source code.

➢ The discussion on maintenance measures will be centered on source code-based measures such as size, complexity, quality, understandability and maintainability.

i. ***Size***: the commonest ways of measuring the size of a program is by counting the number of lines of code.

   ✓ lines of code (LOC) defined as "the count of program lines of code excluding comment or blank lines“

   ✓ This measure is usually expressed in thousands of lines of code (KLOC).

   ✓ During maintenance, the focus is on the 'delta' lines of code: the number of lines of code that have been added or modified during a maintenance process.

Maintenance Measures…

ii. Complexity

- Zuse defines it as "the difficulty of maintaining, changing and understanding programs”
- Program complexity embraces several notions such as program structure, semantic content, control flow, data flow and algorithmic complexity.
- The more complex a program is, the more likely it is for the maintainer to make an error when implementing a change

  a) McCabe's Cyclomatic Complexity

  - McCabe views a program as a directed graph in which lines of program statements are represented by nodes and the flow of control between the statements is represented by the edges.
  - McCabe's cyclomatic complexity (also known as the **cyclomatic number)** is the number of 'linearly independent' paths through the program (or flow graph) and this value is computed using the formula:

    - $v(F) = e - n + 2$

- where n = total number of nodes; e = total number of edges or arcs; and $v(F)$ is the cyclomatic number.

Maintenance Measures…

- Cyclomatic complexity measure is used as an indicator of the psychological complexity of a program.
- During maintenance, a program with a very high cyclomatic number (usually above 10) is considered to be very complex.
- This value can assist the maintainer in a number of ways:
  - To identify highly complex programs that may need to be modified in order to reduce complexity.
  - The *v(F)* can be used as <u>an estimate of the amount of time</u> required to understand and modify a program.
  - The flow graph generated can be <u>used to identify the possible test paths</u> during testing.
- McCabe's cyclomatic number has limitations:
  - It takes no account of the complexity of the conditions in a program, for example multiple use of Boolean expressions, and over-use of flags.
  - In its original form, it failed to take account of the degree of nesting in a program.

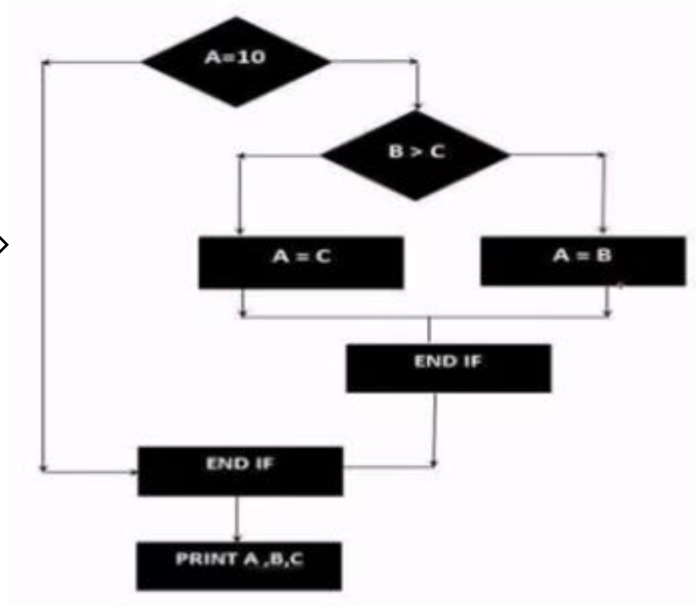
Maintenance Measures…

Cyclomatic complexity measure…

- *v(F) =e-n+2*P*

- where *n* = total number of nodes; *e* = total number of edges or arcs; P= number of nodes that have exit path; and *v(F)* is the cyclomatic number.

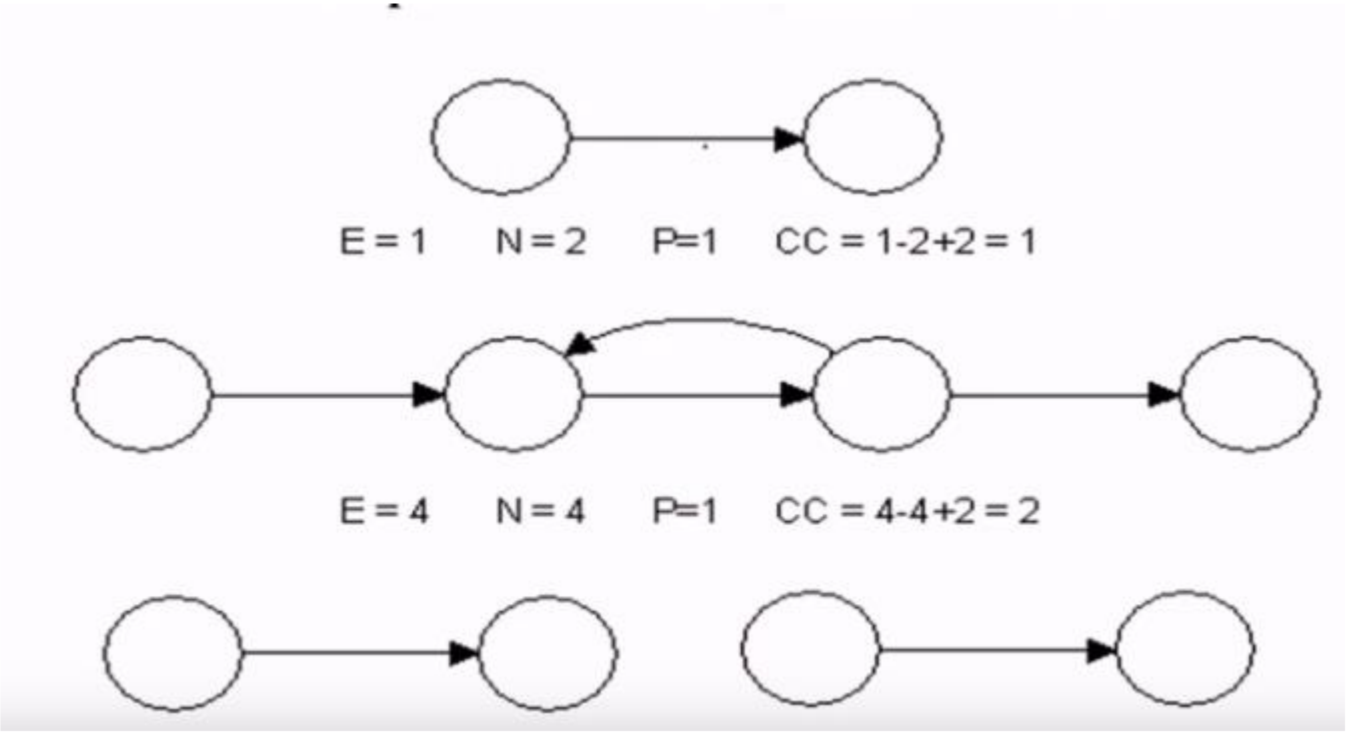
If A= 10 then
If B>C then
A=B
Else A=C
END IF
END IF
Print A, B,C



- N=number of node=7
- E =No of edges(lines)=8
- P=Number of connected component=1

Maintenance Measures…

Examples of Mccabe Number calculations



Maintenance Measures…

- **Halstead's Measures**
- Halstead proposed a number of equations to calculate program attributes such as program length, volume and level, potential volume, language level clarity, implementation time and error rates
- we focus on those measures which impact on complexity: program length & program effort.
- The measures for these attributes can be computed from four basic counts:
  - n1 = number of unique operators used
  - n2 = number of unique operands used
  - *N1* = total number of operators used
  - *N2* = total number of operands used
- An **operand** is a variable or constant.
- An **operator** is an entity that can either change the value of an operand or the order in which it is changed.
- Operators include **arithmetic operators** (for example, *, /, + and -),
- **keywords** (for example, PROCEDURE, WHILE, REPEAT and DO),
- **logical operators** (for example, greater than, equal to and less than), and delimiters.

Maintenance Measures…

- The following formulae can be used to calculate the program length and program effort:
  - Observed program length, $N = N1 + N2;$
  - Calculated program length, = n1log2n1 + n2log2n2
- Program effort, $E = \frac{n_i * N_2 * (N_1 + N2) * \log(n_1 + n_2)}{2 * n_2}$

Advantages of Halstead's Measures

- They are easy to calculate and do not require an in-depth analysis of programming features and control flow.
- The measures can be applied to any language but yet are programming language sensitive.
- There exists empirical evidence from both industry and academia that these measures can be used as good predictors of programming effort and number of bugs in a program.

Maintenance Measures…

Disadvantages of Halstead's Measures

- The experiments which were used to test the measures were badly designed and statistically flawed.
- The counting rules involved in the design of the measures were not fully defined and it is not clear what should be counted.
- There was failure to consider declarations and input/output statements as a unique operator for each unique label

Maintenance Measures…

Halstead's Measures calculations

```
main()
{
int a, b, c, avg;
Scanf ("%d%d%d", &a, &b, &c);
Avg = (a+b+c)/3;
Printf("avg=%d", avg);
}
```

- Size of vocabulary(n=n1+n2): 19
- **Program length(N=N1+N2): 42**
- Program volume: 264
- Program level: 0.04
- **Programming effort: 6000**
- Estimated time: 333 sec

Operator	#
main	1
()	4
{}	1
int	1
scanf	1
&	3
=	1
+	2
/	1
Print	1
,	7
;	4
n1= 12	N1=27

Operand	#
a	3
b	3
c	3
avg	3
"%d%d%d"	1
3	1
"avg=%d"	1
n2= 7	N2=15

Maintenance Measures…

iii. Quality

- In general terms, quality is defined as 'fitness for purpose'.
- In other words, a quality product, be it a word processor or a flight control system, is one which does what the user expects it to do.
- A quality maintenance process is one which enables the maintainer to implement the desired change.

a. Product Quality

- One way of measuring the quality of a software system is by keeping track of the number of change requests received from the users after the system becomes operational.
- This measure is computed:

$$PQ = \frac{UCR}{TKLOC}$$

- where UCR = number of unique change requests made by customers for the first year of field use of a given release,
- TKLOC= the number of thousand lines of code for that release
- PQ=Product Quality

Maintenance Measures…

b. Process Quality

- This describes the degree to which the maintenance process being used is assisting personnel in satisfying change requests.
- Two measures of process quality are **schedule** and **productivity**.

1) The **schedule** is calculated as "the difference between the planned and actual work time to achieve the milestone of first customer delivery, divided by the planned work time".

- This measure is expressed as a percentage. A negative number signifies a slip and a positive number signifies early delivery.

2) The **productivity** is computed by dividing the number of lines of code that have been added or modified by the effort in staff days required to make the addition or modification.

- Effort is the total time from analyzing the change requests to a successful implementation of the change.

Maintenance Measures…

iv. Understandability

- Program understandability is the ease with which the program can be understood, that is, the ability to determine what a program does and how it works by reading its source code and accompanying documentation.
- This attribute depends not just on the program source code, but also on other external factors such as the available **documentation**, the maintenance **process** and maintenance **personnel**.
- Understandability usually has an inverse relation to complexity; *as the complexity of a program increases, the understandability tends to decrease.*
- From this perspective, understandability can be computed indirectly from McCabe's cyclomatic complexity and Halstead's program effort measure.

Maintenance Measures…

v. Maintainability

- Software maintainability is ***the ease with which the software can be understood, corrected, adapted, and/or enhanced***.
- ***Maintainability*** is an external attribute since its computation requires knowledge from the software product as well as external factors such as the maintenance process and the maintenance personnel.
- An example of a *maintainability* measure that depends on an external factor is the Mean Time To Repair (MTTR): the mean time required to effect a change.
- Depending on the circumstances, the calculation of ***MTTR*** may require information on the *problem recognition time*, *administrative delay time*, *maintenance tools collection time*, *problem analysis time*, *change specification time* and *change time*.
- MTTR = total repair time / total repairs

vi. Cost Estimation

- The cost of a maintenance project is the resources - *personnel*, *machines*, *time* and *money* - expended on effecting change.
- One way of estimating the cost of a maintenance task is from *historical data collected* for a similar task.
- The major difficulty with this approach to cost estimation is that there may be new variables impacting upon the current task which were not considered in the past.
- A second way of estimating cost is through *mathematical models*.
- One of these was Boehm's *COCOMO model* adapted for maintenance.
- The updated COCOMO II model
- According to Boehm, the cost of maintenance is affected by attributes of factors called *cost drivers.*
- Examples of ***cost drivers*** are database size, program complexity, use of modern programming practices and applications experience of the maintenance personnel.

Guidelines For Selecting Maintenance Measures

- The main purpose of maintenance activities is to ensure that a software system can be easily modified, adapted and enhanced to accommodate changes.
- There are no hard and fast rules as to how these objectives can be achieved through the use of maintenance measures.
- some guidelines that can be used in selecting suitable maintenance measures.

i. **Clearly defined objectives:**

- Prior to deciding on the use of a measurement for maintenance-related purposes, it is essential to define clearly and unambiguously what objectives need to be achieved.
- These objectives will determine the measures to be used and the data to be collected.

ii. **Personnel involvement:**

- The purpose of measurement in an organisation needs to be made clear to those involved in the programme.
- And the measures obtained should be used for that purpose and nothing else.
- For instance, it needs to be made clear whether the measurement is to improve productivity, to set and monitor targets, etc.

Guidelines…

iii. Ease of use:

- The measures that are finally selected to be used need to be easy to use, take not too much time to administer, be unobtrusive, and possibly subject to automation.

Reference

- ✓ Penny Grub, Armstrong A Takang, Software Maintenance Concepts and Practice, 2nd edition
- ✓ Alain April, Alain Abran (2008), Software Maintenance Management Evaluation and Continuous Improvement.
- ✓ Pierre Bourque, École de technologie supérieure(2014), Guide to the Software Engineering Body of Knowledge (SWEBOK) Version 3.0, A Project of the IEEE Computer Society.