SOFTWARE MAINTENANCE ASSIGNMENT

- ASHISH KUMAR
- 2K18/SE/041

Ques: Identify, list, and briefly explain (at least 6) the software maintenance measures from online SCI-indexed articles (IEEE, Science Direct, Springer, Wiley, etc).

Note: Ma'am I have searched a lot on internet, but I am able to find only 1 article on software maintenance measures from IEEE (link: https://ieeexplore.ieee.org/document/1702781) and it contains that measures (logical stability measures) which are not same as the measures that is written in Software maintenance Book. So, ma'am I am writing the measures that is written in the book and which was taught by you in class.

Ans: The 6 software maintenance measures are:

1. **Size**

One of the most common ways of measuring the size of a program is by counting the number of lines of code. Moller and Paulish define lines of code (LOC) 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. The advantage of this measure is that it is easy to determine and also correlates strongly with other measures such as effort and error density. There are no standards for LOC measurement and it is dependent on the programming language in question. Also it is too simplistic and does not reflect cost or productivity.

2. **Complexity**

There is no consensus on a definition for the term 'complexity'. In software maintenance, however, it is useful to define program complexity. Zuse defines it as "the difficulty of maintaining, changing and understanding programs". One of the major problems that software maintainers face is dealing with the increasing complexity of the source code that they have to modify. 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. The higher the complexity value, the more difficult it is to understand the program, hence making it less maintainable.

Two of the most popular code complexity measures: McCabe's cyclomatic complexity and Halstead's difficulty measure:

- **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. This 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:

- ✓ It helps to identify highly complex programs that may need to be modified in order to reduce complexity.
- ✓ The cyclomatic number can be used as an estimate of the amount of time required to understand and modify a program.
- ✓ The flow graph generated can be used to identify the possible test paths during testing.

- **Halstead's Measures**

In his theory of software science, 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. Here we shall concentrate on those measures which impact on complexity: program length and 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

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 = n1*N2*(N1+N2)*log(n1+n2) / 2*n2

The four basic counts can also be used to compute other attributes such as programming time, number of bugs in a program and understandability.

There are a number of reasons why Halstead's measures have been widely used. Firstly, they are easy to calculate and do not require an in-depth analysis of programming features and control flow. Secondly, the measures can be applied to any language but yet are programming language sensitive.

3. 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. Different measures can be used to characterize product and process quality.

- **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 by "dividing the number of unique change requests made by customers for the first year of field use of a given release, by the number of thousand lines of code for that release". This measure only includes requests which pertain to faults detected by customers.

The other measure of product quality is the number of faults that are detected after the software system becomes operational, usually after the first year of shipment. The same type of fault pointed out by more than one user is counted as a single fault. The number of users reporting the same fault may be used as a measure of the significance of the fault and therefore the priority that should be attached to fixing it.

- **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. 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.

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.

4. 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. Some of the measures that can be used as an estimate of understandability are complexity, quality of documentation, consistency and conciseness.

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.

5. 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.

Maintainability can also be perceived as an internal attribute if it is derived solely from the software system. Several internal attributes of program source code can impact on maintainability, for example modularity. Thus, measures for these other internal attributes

would need to be obtained in order to compute maintainability. At present, measures such as complexity and readability are used as indicators or predictors of maintainability.

6. 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. However, the more that is collected the more accurate will be the estimate.

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, instead of being based on a single process model such as the waterfall model, has been interpreted to cover the waterfall model, MBASE/RUP (Model- Based Architecting and Software Engineering / Rational Unified Process), and incremental development.

A third measure of cost is time in person-months required to modify a program.

In table format:

Measure	Definition	Formula(if any)
1. Size	One of the most common ways of measuring the size of a program is by counting the number of lines of code. Moller and Paulish define lines of code (LOC) as "the count of program lines of code excluding comment or blank lines".	N/A
2. 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 higher the complexity value, the more difficult it is to understand the program, hence making it less maintainable.	N/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)** 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.
• **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.	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 = n1*N2*(N1+N2)*log(n1+n2) / 2*n2
3. 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. Quality can be characterized into product and process quality.	N/A
4. 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	N/A

	documentation. Some of the measures that can be used as an estimate of understandability are complexity, quality of documentation, consistency and conciseness. Understandability usually has an inverse relation to complexity; as the complexity of a program increases, the understandability tends to decrease.	
5. 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. Maintainability can also be perceived as an internal attribute if it is derived solely from the software system.	N/A
6. Cost Estimation	The cost of a maintenance project is the resources - personnel, machines, time and money expended on effecting change.	N/A