**STQA**

**ASSIGNMENT NO. 2**

# **Q1. What is the measurement unit of Halstead’s Programming Effort (E)?**

Halstead’s Programming Effort: Halstead’s Programming Effort measures the amount of mental activity needed to translate the existing algorithm into implementation in the specified program language. The unit of measurement of E is elementary mental discriminations.

*E = V / L = D \* V = Difficulty \* Volume*

Where,

V = Program Volume

L = Program Level

D = Program Difficulty

# **Q2. What are the base measures in all of Halstead metrics?**

Halstead’s Metrics: Halstead’s metrics are included in a number of current commercial tools that count software lines of code. By counting the tokens and determining which are operators and which are operands, the following base measures can be collected:

* n1 = Number of distinct operators.
* n2 = Number of distinct operands.
* N1 = Total number of occurrences of operators.
* N2 = Total number of occurrences of operands.

In addition to the above, Halstead defines the following :

* n1\* = Number of potential operators.
* n2\* = Number of potential operands.

Halstead metrics are :

Halstead Program Length: The total number of operator occurrences and the total number of operand occurrences.

N = N1 + N2

And estimated program length is, *N^ = n1log2n1 + n2log2n2*

The following alternate expressions have been published to estimate program length:

* NJ = log2(n1!) + log2(n2!)
* NB = n1 \* log2n2 + n2 \* log2n1
* NC = n1 \* sqrt(n1) + n2 \* sqrt(n2)
* NS = (n \* log2n) / 2

Halstead Vocabulary: The total number of unique operator and unique operand occurrences.

*n = n1 + n2*

Program Volume: Proportional to program size, represents the size, in bits, of space necessary for storing the program. This parameter is dependent on specific algorithm implementation. The properties V, N, and the number of lines in the code are shown to be linearly connected and equally valid for measuring relative program size.

*V = size \* (log2 vocabulary) = N \* log2(n)*

The unit of measurement of volume is the common unit for size “bits”. It is the actual size of a program if a uniform binary encoding for the vocabulary is used. And error = Volume / 3000

Potential Minimum Volume: The potential minimum volume V\* is defined as the volume of the most succinct program in which a problem can be coded.

*V\* = (2 + n2\*) \* log2(2 + n2\*)*

Here, n2\* is the count of unique input and output parameters

Program Level: To rank the programming languages, the level of abstraction provided by the programming language, Program Level (L) is considered. The higher the level of a language, the less effort it takes to develop a program using that language.

*L = V\* / V*

The value of L ranges between zero and one, with L=1 representing a program written at the highest possible level (i.e., with minimum size).

And estimated program level is *L^ =2 \* (n2) / (n1)(N2)*

Program Difficulty: This parameter shows how difficult to handle the program is.

*D = (n1 / 2) \* (N2 / n2)*

*D = 1 / L*

As the volume of the implementation of a program increases, the program level decreases and the difficulty increases. Thus, programming practices such as redundant usage of operands, or the failure to use higher-level control constructs will tend to increase the volume as well as the difficulty.

Programming Effort: Measures the amount of mental activity needed to translate the existing algorithm into implementation in the specified program language.

*E = V / L = D \* V = Difficulty \* Volume*

Language Level: Shows the algorithm implementation program language level. The same algorithm demands additional effort if it is written in a low-level program language. For example, it is easier to program in Pascal than in Assembler.

*L’ = V / D / D*

*= L \* V\* = L2 \* V*

Intelligence Content: Determines the amount of intelligence presented (stated) in the program This parameter provides a measurement of program complexity, independently of the program language in which it was implemented.

*I = V / D*

Programming Time: Shows time (in minutes) needed to translate the existing algorithm into implementation in the specified program language.

*T = E / (f \* S)*

The concept of the processing rate of the human brain, developed by the psychologist John Stroud, is also used. Stroud defined a moment as the time required by the human brain to carry out the most elementary decision. The Stroud number S is therefore Stroud's moments per second with 5 <= S <= 20.

Halstead uses 18. The value of S has been empirically developed from psychological reasoning, and its recommended value for programming applications is 18.

*Stroud number S = 18 moments / second*

*seconds-to-minutes factor f = 60*

# **Q3. Consider the following values and calculate the value of Halstead’s Programming Effort (E)**

| n1 = 14 | N1 = 53 | n2 = 10 | N2 = 38 |
| --- | --- | --- | --- |
| n1 = 17 | N1 = 45 | n2 = 12 | N2 = 32 |
| n1 = 26 | N1 = 48 | n2 = 23 | N2 = 56 |

Taking values:

* n1 = 14
* N1 = 53
* n2 = 10
* N2 = 38

Program Vocabulary:

n = n1 + n2 = 14 + 10 = 24

Program Length:

N = N1 + N2 = 53 + 38 = 91

Program Volume:

V = Nlog2n = 91log224 = 417.2315876

Program Level:

PL = 1 / [(n1/2) \* (N2/n2)] = 1 / [(14/2) \* (38/10)] = 0.03759398496

Halstead’s Programming Effort:

e = V / PL = 417.2315876 / 0.03759398496 = 11098.36023

Taking values:

* n1 = 17
* N1 = 45
* n2 = 12
* N2 = 32

Program Vocabulary:

n = n1 + n2 = 17 + 12 = 29

Program Length:

N = N1 + N2 = 45 + 32 = 77

Program Volume:

V = Nlog2n = 77log229 = 374.0645366

Program Level:

PL = 1 / [(n1/2) \* (N2/n2)] = 1 / [(17/2) \* (32/12)] = 0.04411764706

Halstead’s Programming Effort:

e = V / PL = 374.0645366 / 0.04411764706 = 8478.796163

Taking values:

* n1 = 26
* N1 = 48
* n2 = 23
* N2 = 56

Program Vocabulary:

n = n1 + n2 = 26 + 23 = 49

Program Length:

N = N1 + N2 = 48 + 56 = 104

Program Volume:

V = Nlog2n = 104log249 = 583.9298238

Program Level:

PL = 1 / [(n1/2) \* (N2/n2)] = 1 / [(26/2) \* (56/23)] = 0.03159340659

Halstead’s Programming Effort:

e = V / PL = 583.9298238 / 0.03159340659 = 18482.64834

# **Q4. Compute the function point, productivity, documentation, cost per function for the following data:**

1. Number of user inputs = 24
2. Number of user outputs = 46
3. Number of inquiries = 8
4. Number of files = 4
5. Number of external interfaces = 2
6. Effort = 36.9 p-m
7. Technical documents = 265 pages
8. User documents = 122 pages
9. Cost = $7744/ month

Various processing complexity factors are: 4, 1, 0, 3, 3, 5, 4, 4, 3, 3, 2, 2, 4, 5.

| Measurement Parameter | Count | Weighing Factor |
| --- | --- | --- |
| Number of external inputs (EI) | 24 | 24 \* 4 = 96 |
| Number of external outputs (EO) | 46 | 46 \* 4 = 184 |
| Number of external inquiries (EQ) | 8 | 8 \* 6 = 48 |
| Number of internal files (ILF) | 4 | 4 \* 10 = 40 |
| Number of external interfaces (EIF) Count | 2 | 2 \* 5 = 10 |
| Total | 84 | 372 |

So sum of all fi (i from 1 to 14) = 4 + 1 + 0 + 3 + 5 + 4 + 4 + 3 + 3 + 2 + 2 + 4 + 5 = 43

Function Point = Count - Total \* [0.65 + 0.01 \* ∑(fi)] = 378 \* [0.65 + 0.01 \* 43] = 408.24

Productivity = FP / Effort = 408.24 / 36.9 = 11.06341463

Total pages of documentation = Technical Document + User Document = 265 + 122 = 387 pages

Documentation = Pages of documentation / FP = 387 / 408.24 = 0.9479717813

Cost per Function = Cost / Productivity = 7744 / 11.06341463 = 699.9647269 (approx 700)

# **Q5. Consider a project with the following distribution of data and calculate its defect spoilage.**

| SDLC Phase | No. of Defects | Defect Age |
| --- | --- | --- |
| Requirement Specifications | 34 | 2 |
| HLD | 25 | 4 |
| LLD | 17 | 5 |
| Coding | 10 | 6 |

Spoilage = Sum of (Number of Defects \* Defect Age) / Total number of Defects = (34 \* 2 + 25 \* 4 + 17 \* 5 + 10 \* 6) / 86 = 3.639534884