Software Metrics

Software Metrics

Software Metrics: What and Why ?

1. How to measure the size of a software?
2. How much will it cost to develop a software?
3. How many bugs can we expect?
4. When can we stop testing?
5. When can we release the software?

Software Metrics

6. What is the complexity of a module?

7. What is the module strength and coupling?

8. What is the reliability at the time of release?

9. Which test technique is more effective?

10. Are we testing hard or are we testing smart?

11. Do we have a strong program or a week test suite?

Software Metrics

- Pressman explained as "A measure provides a quantitative indication of the extent, amount, dimension, capacity, or size of some attribute of the product or process".
- Measurement is the act of determine a measure
- The metric is a quantitative measure of the degree to which a system, component, or process possesses a given attribute.
- Fenton defined measurement as " it is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules".

Software Metrics

- Definition

Software metrics can be defined as "*The continuous application of measurement based techniques to the software development process and its products to supply meaningful and timely management information, together with the use of those techniques to improve that process and its products*".

Software Metrics

- Areas of Applications

The most established area of software metrics is cost and size estimation techniques.

The prediction of quality levels for software, often in terms of reliability, is another area where software metrics have an important role to play.

The use of software metrics to provide quantitative checks on software design is also a well established area.

Software Metrics

- Problems During Implementation

➢ Statement	: Software development is to complex; it cannot be managed like other parts of the organization.
Management view	: Forget it, we will find developers and managers who will manage that development.
➢ Statement	: I am only six months late with project.
Management view	: Fine, you are only out of a job.

Software Metrics

➢ Statement	: I am only six months late with project.
Management view	: Fine, you are only out of a job.
➢ Statement	: But you cannot put reliability constraints in the contract.
Management view	: Then we may not get the contract.

Software Metrics

- Categories of Metrics

  i. **Product metrics:** describe the characteristics of the product such as size, complexity, design features, performance, efficiency, reliability, portability, etc.

  ii. **Process metrics:** describe the effectiveness and quality of the processes that produce the software product. Examples are:

  - effort required in the process
  - time to produce the product
  - effectiveness of defect removal during development
  - number of defects found during testing
  - maturity of the process

Software Metrics

ii. **Project metrics:** describe the project characteristics and execution. Examples are :

- number of software developers
- staffing pattern over the life cycle of the software
- cost and schedule
- productivity

Software Metrics

Token Count

The size of the vocabulary of a program, which consists of the number of unique tokens used to build a program is defined as:

$$\eta = \eta_1 + \eta_2$$

where

η : vocabulary of a program

η_1 : number of unique operators

η_2 : number of unique operands

Software Metrics

The length of the program in the terms of the total number of tokens used is

$$N = N_1 + N_2$$

where

N : program length

N_1 : total occurrences of operators

N_2 : total occurrences of operands

Software Metrics

Volume

$$V = N * \log_2 \eta$$

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.

Program Level

$$L = V^* / V$$

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).

Software Metrics

Program Difficulty

$$D = 1 / L$$

As the volume of an 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.

Effort

$$E = V / L = D * V$$

The unit of measurement of E is elementary mental discriminations.

Software Metrics

- Estimated Program Length

$$\hat{N} = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2$$

$$\hat{N} = 14 \log_2 14 + 10 \log_2 10$$

$$= 53.34 + 33.22 = 86.56$$

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

$$N_J = Log_2(\eta_1!) + \log_2(\eta_2!)$$

Software Metrics

$$N_B = \eta_1 Log_2 \eta_2 + \eta_2 \log_2 \eta_1$$

$$N_c = \eta_1 \sqrt{\eta_1} + \eta_2 \sqrt{\eta_2}$$

$$N_s = (\eta \log_2 \eta) / 2$$

The definitions of unique operators, unique operands, total operators and total operands are not specifically delineated.

Software Metrics

- Counting rules for C language

1. Comments are not considered.
2. The identifier and function declarations are not considered.
3. All the variables and constants are considered operands.
4. Global variables used in different modules of the same program are counted as multiple occurrences of the same variable.

Software Metrics

5. Local variables with the same name in different functions are counted as unique operands.

6. Functions calls are considered as operators.

7. All looping statements e.g., do {…} while ( ), while ( ) {…}, for ( ) {…}, all control statements e.g., if ( ) {…}, if ( ) {…} else {…}, etc. are considered as operators.

8. In control construct switch ( ) {case:…}, switch as well as all the case statements are considered as operators.

Software Metrics

9. The reserve words like return, default, continue, break, sizeof, etc., are considered as operators.

10. All the brackets, commas, and terminators are considered as operators.

11. GOTO is counted as an operator and the label is counted as an operand.

12. The unary and binary occurrence of “+” and “-” are dealt separately. Similarly “*” (multiplication operator) are dealt with separately.

Software Metrics

13. In the array variables such as “array-name [index]” “array-name” and “index” are considered as operands and [ ] is considered as operator.

14. In the structure variables such as “struct-name, member-name” or “struct-name -> member-name”, struct-name, member-name are taken as operands and ‘.’, ‘->’ are taken as operators. Some names of member elements in different structure variables are counted as unique operands.

15. All the hash directive are ignored.

Software Metrics

- Potential Volume

$$V^* = (2+\eta_2^*)\log_2(2+\eta_2^*)$$

- Estimated Program Level / Difficulty

Halstead offered an alternate formula that estimate the program level.

$$\hat{L} = 2\eta_2 / (\eta_1 N_2)$$

where

$$\hat{D} = \frac{1}{\hat{L}} = \frac{\eta_1 N_2}{2\eta_2}$$

Software Metrics

- Effort and Time

$$E = V / \hat{L} = V * \hat{D}$$

$$= (n_1 N_2 N \log_2 \eta) / 2\eta_2$$

$$T = E / \beta$$

β is normally set to 18 since this seemed to give best results in Halstead's earliest experiments, which compared the predicted times with observed programming times, including the time for design, coding, and testing.

Software Metrics

- Language Level

$$\lambda = L \times V^* = L^2 V$$

Using this formula, Halstead and other researchers determined the language level for various languages as shown in Table 1.

Software Metrics

Language	*Language Level λ*	*Variance σ*
PL/1	1.53	0.92
ALGOL	1.21	0.74
FORTRAN	1.14	0.81
CDC Assembly	0.88	0.42
PASCAL	2.54	–
APL	2.42	–
C	0.857	0.445

Table 1: Language levels

Software Metrics

Example- 6.I

Consider the sorting program in Fig. 2 of chapter 4. List out the operators and operands and also calculate the values of software science measures like η, N, V, E, λ $etc.$

Software Metrics

Solution

The list of operators and operands is given in table 2.

Operators	*Occurrences*	*Operands*	*Occurrences*
int	4	SORT	1
( )	5	x	7
,	4	n	3
[ ]	7	i	8
if	2	j	7
<	2	save	3

(Contd.)...

Software Metrics

;	11	im1	3
for	2	2	2
=	6	1	3
–	1	0	1
< =	2	—	—
+ +	2	—	—
return	2	—	—
{ }	3	—	—
$\eta_1 = 14$	$N_1 = 53$	$\eta_2 = 10$	$N_2 = 38$

Table 2: Operators and operands of sorting program of fig. 2 of chapter 4

Software Metrics

Here N_1=53 and N_2=38. The program length $N=N_1+N_2=91$

Vocabulary of the program $\eta = \eta_1 + \eta_2 = 14 + 10 = 24$

Volume $V = N \times \log_2 \eta$

$= 91 \times \log_2 24 = 417$ bits

The estimated program length $\hat{N}$ of the program

$= 14 \log_2 14 + 10 \log_2 10$

$= 14 * 3.81 + 10 * 3.32$

$= 53.34 + 33.2 = 86.45$

Software Metrics

Conceptually unique input and output parameters are represented by η_2^*

$\eta_2^* = 3$ {x: array holding the integer to be sorted. This is used both as input and output}.

{N: the size of the array to be sorted}.

The potential volume $V^* = 5 \log_2 5 = 11.6$

Since $L = V^* / V$

Software Metrics

$$= \frac{11.6}{417} = 0.027$$

D = I / L

$$= \frac{1}{0.027} = 37.03$$

Estimated program level

$$\hat{L} = \frac{2}{\eta_1} \times \frac{\eta_2}{N_2} = \frac{2}{14} \times \frac{10}{38} = 0.038$$

Software Metrics

We may use another formula

$$\hat{V} = V \times \hat{L} = 417 \times 0.038 = 15.67$$

$$\hat{E} = V / \hat{L} = \hat{D} \times V$$

$= 417 / 0.038 = 10973.68$

Therefore, 10974 elementary mental discrimination are required to construct the program.

$$T = E / \beta = \frac{10974}{18} = 610 \,\text{seconds} = 10 \,\text{minutes}$$

This is probably a reasonable time to produce the program, which is very simple

Software Metrics

```
#include < stdio.h >
#define MAXLINE 100
int getline(char line[],int max);
int strindex(char source[],char search for[]);
char pattern[ ]="ould";
int main()
{
    char line[MAXLINE];
    int found = 0;
    while(getline(line,MAXLINE)>0)
        if(strindex(line, pattern)>=0)
        {
            printf("%s",line);
            found++;
        }
    return found;
}
```

Table 3

(Contd.)...

Software Metrics

```
    int getline(char s[],int lim)
    {
        int c,i=0;
        while(--lim > 0 && (c=getchar())!= EOF && c!=‘\n’)
            s[i++]=c;
        if(c==‘\n’)
            s[i++] = c;
        s[i] = ‘\0’;
        return i;
    }
    int strindex(char s[],char t[])
    {
        int i,j,k;
        for(i=0;s[i] !='\0';i++)
        {
            for(j=i,k=0;t[k] != '\0',s[j] ==t[k];j++,k++);
            if(k>0 && t[k] =='\0')
                return i;
        }
        return -1;
    }
```

Table 3

Software Metrics

Example- 6.2

Consider the program shown in Table 3. Calculate the various software science metrics.

Software Metrics

Solution

List of operators and operands are given in Table 4.

Operators	*Occurrences*	*Operands*		*Occurrences*
main ()	1	—		—
–	1	**Extern variable**	pattern	1
for	2	**main function**	line	3
= =	3		found	2
! =	4	**getline function**	s	3
getchar	1		lim	1

Table 4

(Contd.)...

Software Metrics

( )	1	c	5
&&	3	i	4
--	1	**Strindex function** s	2
return	4	t	3
++	6	i	5
printf	1	j	3
>=	1	k	6
strindex	1	**Numerical Operands** 1	1
If	3	MAXLINE	1
>	3	0	8
getline	1	'\0'	4
while	2	'\n'	2
{ }	5	**strings "ould"**	1
=	10	—	—
[ ]	9	—	—
,	6	—	—
;	14	—	—
EOF	1	—	—
$\eta_1 = 24$	$N_1 = 84$	$\eta_2 = 18$	$N_2 = 55$

Table 5

Software Metrics

Program vocabulary $\eta = 42$

Program length $N = N_1 + N_2$

$= 84 + 55 = 139$

Estimated length $\hat{N} = 24\log_2 24 + 18\log_2 18 = 185.115$

% error $= 24.91$

Program volume $V = 749.605$ bits

Estimated program level $= \frac{2}{\eta_1} \times \frac{\eta_2}{N_2}$

$$= \frac{2}{24} \times \frac{18}{55} = 0.02727$$

Software Metrics

Minimal volume $V^*=20.4417$

Effort $= V / \hat{L}$

$$= \frac{748.605}{.02727}$$

= 27488.33 elementary mental discriminations.

Time T = $E / \beta = \frac{27488.33}{18}$

= 1527.1295 seconds

= 25.452 minutes