Correlations and Weights of Maintainability Index (MI) of Open source Linux Kernel Modules

Mohamed B. Senousy
Professor of Computer Science
Sadaat Academy
Computer Science and Information Technology, Cairo, Egypt

Tamer Sh. Mazen
Lecturer assistant, Dept. of Management Information Systems, Modern Academy for Computer Science and Information Technology, Cairo, Egypt

ABSTRACT
Software development and usage become very important in many aspects of our lives, so that the application of software metrics becomes more important. Software metrics are used to give valuable information for the development of software. This paper focus on the study of the maintainability of an open source operating system "Linux". Computing the relative weight of each maintainability parameter. The research was conducted on Linux Kernel modules (V.3.9.2). The research performed on 837 functions from selected modules to compute a maintainability index "MI" for each function and module. Also calculate the correlation between each parameter for Maintainability Index (MI) with the other parameters and with the MI itself. The parameters of MI are Line of Code (LOC), Cyclomatic Complexity (CC), and Halstead Volume (HV). Practically approved that, first the Line of Code and Cyclomatic Complexity values are distributed normally; but Halstead Volume is distributed uniformly. Secondly, there is an interconnection between Line of Code, Cyclomatic Complexity, Halstead Volume and Maintainability Index. Finally the most important parameter which affects maintainability index is Line of Code, then Cyclomatic Complexity and lastly the Halstead Volume.

General Terms
Software Quality

Keywords
Maintainability Index (MI), Analytic Hierarchy Process (AHP), Correlation coefficient, Chi square test

1. INTRODUCTION
Quality is one of the terms that are simple but at the same time difficult to be defined exactly. The quality of the product means how well this product performs in its use. This includes all the characteristics of this product [1] [2].

According to the growth in demand of software systems, the need for high quality and efficient software systems has also increased. The quality of software can be measured as any product through its attributes. The main attributes of software quality are usability, reliability, portability, and maintainability [3] [4].

Software maintainability is the degree with which changes and modifications can be made easily to a software system. Maintainability is highly related to the maintenance of a software system as it can be also defined as easiness to perform maintenance of the system [5].

One of the ways used to quantify the software maintainability is the Maintainability Index (MI). The MI is a combination of software metrics called Lines of Code (LOC), Cyclomatic Complexity (CC), and Halstead Volume (HV) [4].

Line of Code is a software metric used to measure the size of a computer program by counting the number of lines in the text of the program's source code [6]. CC which has been introduced by Thomas McCabe is a measure of the number of linearly-independent paths through a program module (Control Flow).

Halstead Volume is a software metric introduced by Maurice Howard Halstead in 1977 as part of his treatise on establishing an empirical science of software development. Halstead makes the observation that metrics of the software should reflect the implementation or expression of algorithms in different languages, but be independent of their execution on a specific platform [7][8].

[7][9] Proved that MI values above 85 indicate a highly maintenance software. Values between 65 and 85 reflect a moderate maintainability while values below 65 indicate that it is difficult to maintain.

Ilja Heritage [7], based on ISO9126 model for software product quality discusses several problems with the maintainability index and identify a number or requirement to be fulfilled by a maintainability model to be usable in practical, they draw a new maintainability model that solve most of these problems.

Anita Ganpati et al. [6] proposed the maintainability index observed over fifty successive versions. Applied on Apache, Mozilla Firefox, MySql and FileZilla software. Calculated software metrics using resource standard metrics tool and Crystal Flow tool. Resulted that maintainability index was highest in case of Mozilla Firefox and was lowest in the case of Apache OSS.

Dimitris Stavrinoudis et al. [10] proposed the relation between software metrics and maintainability metrics which characterize the ease of the maintenance process when applied to a specific product. Setting up measurement and metrics standers help block failures before the maintenance process and reduce the necessary effort.

Meine and Revilla [11] proposed the relation between internal and external software metrics. Their experiment shows that, there are a correlation between line of code, Halstead Volume and Cyclomatic Complexity.

Chauhan and Sharma [12] focused on predicting the maintainability of two open source software. They studied the impact of software metrics Lines of code, Cyclomatic Complexity and Halstead volume over maintainability using "JHawk" tool. The result illustrated that these metrics have strong composite impact over the maintainability of open source software due to involvement of human and environmental factors. However most of these researches focused on the pre-paid software and very little work done on open source software.

The main contribution of this work is that maintainability of open source basic software (Linux) is calculated. Three internal software metrics LOC, CC and HV are used to calculate the maintainability index. Also, the Chi square test and the Analytical Hierarchy Process (AHP) are used to test the distribution and the weights for each of the MI parameters in sequence. The AHP is calculated based on the results of the correlation between the MI parameters.

2. BACKGOUND

In this research applying some statistical and probabilistic approach due to the random nature of the variables affected the quality metrics. It is obvious without proof that maintainability index well depends on Chi square test, correlation coefficient and analytical hierarchy process. So in the following a review for these concepts will be given.

2.1 Chi Square Test

Chi square test used to test the distribution and the weights for each of the MI parameters in sequence [13][14][15].

For computing Chi square test, by suppose two hypotheses H_0 as the null hypothesis and H_1 as the alternative hypothesis.

The chi square test uses following formula

$$X^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \quad \text{.....(1)}$$

$$E = \frac{\sum_{i=1}^{n}(O_i)}{\text{Count } (O)} \quad \text{.....(2)}$$

Where:

n is the number of elements,

X^2 is chi square result,

O is the Observed Frequency in each category,

And E is the Expected Frequency in the corresponding category.

After calculating chi square for Observed Frequency, search for tabular chi square in freedom table after calculating freedom by the following equation [13][14][15]:

$$Df = (\text{Count } (O) - 1) \quad \text{.....(3)}$$

Where:

O is the same as before.

The last step is to compare the calculated chi square result vs. tabular chi square result, if the calculated result is greater than tabular result then approved H_1 hypothesis else H_0 is approved [13][14][15].

2.2 Correlation coefficient

A correlation coefficient is a statistical measure of the degree to which changes to the value of one variable predict change to the value of another. In positively correlated variables, the value increases or decreases in tandem. In negatively correlated variables, the value of one increases as the value of the other decreases [15].

Correlation coefficients are expressed as values between +1 and -1. A coefficient of +1 indicates a perfect positive correlation: A change in the value of one variable will predict a change in the same direction in the second variable. A coefficient of -1 indicates a perfect negative correlation: A change in the value of one variable predicts a change in the opposite direction in the second variable. Lesser degrees of correlation are expressed as non-zero decimals. A coefficient of zero indicates there is no discernible relationship between fluctuations of the variables [15].

There are some steps to calculate correlation coefficient. The data which working with are paired data, each pair of which will be denoted by (xi, yi).

The quantities from these calculations will be used in subsequent steps of calculation. First, the mean of the entire first and second coordinate data Xi & Yi is calculated. Then, calculated the standard deviation of first and second coordinates of the data Xi & Yi [15].

Using following equation to calculate correlation coefficient

$$r = \frac{n(\sum XY) - (\sum X)(\sum Y)}{\sqrt{[n\sum X^2 - (\sum X^2)][n\sum Y^2 - (\sum XY^2)]}} \quad \text{.....(4)}$$

2.3 Analytical Hierarchy Process Calculation

The analytic hierarchy process (AHP) is a structured technique for organizing and analyzing complex decisions, based on mathematics and psychology, It is a multi-criteria decision-making approach [16] [17].

To compute AHP, there are four steps. First step: estimation of the pertinent data, but estimated in matrix A. It has relied on Intensity of Importance for parameters.

Table 1 illustrates Intensity of Importance Scale according to Saaty [16] [17]

Table 1: Intensity of the Importance Scale

Intensity of Importance	Definition
1	Objectives *i* and *j* are of equal importance.
3	Objective *i* is weakly more important than *j*.
5	Objective *i* is strongly more important than *j*.
7	Objective *i* is very strongly more important than *j*.
9	Objective *i* is absolutely more important than *j*.
2, 4, 6, 8	Intermediate values.

The second step is copying matrix A to matrix B. The third step is squaring the matrix by multiplying matrix A by B and fourth step is computing the eigenvector.

3. RRESEARCH EXPERIEMENT

3.1 Module Separation

As mentioned before dealing with Linux kernel(V.3.9.2). First separate the software into modules and calculate the Maintainability Index (MI) for each module. To calculate the MI for each module, separate the module itself into its components (functions). Figure 1 shows the pseudo-code for module separation into its functions.

```
Open module file as text
While not end of file
{
Read line
If line start by functions definitions
        {
         Read next line
         If next line =”{“
            {
       Call get_function_name()
       Push “{“ in stack
       While stack not empty
      {
      Read next file
        If next line =”}”
         {
          Pop from stack
         Else
          Add line in linked list
         }
      }

       }
    }
  }
```

Figure 1: Pseudo-code for separation module

3.2 Maintainability Index Calculation

3.2.1 Maintainability Index for Functions

Function is a part of the module. First needs to calculate the MI for each function separately. Maintainability Index (MI) is calculated by using the following equation [5][8][18].

$$MI = 171 - 5.2\ln(HV) - 0.23(CC) - 16.2\ln(LOC) \quad \ldots..(5)$$

Where:

HV is series of tokens which can be classified as Operators (any symbol or reserved keyword in a program that specifies an algorithmic action, most punctuation marks) and Operands (any symbol used to represent data).

$$CC = \#\text{condition statments} + \#\text{loops statments} + 1. \quad (6)$$

LOC is the number of lines of code in the function.

Table 2 represents sample of the results of the calculated MI for the Linux kernel functions.

Table 2: Sample of 25 functions MI

Serial	Function Name	Maintainability Index
1	check_free_space	80%
2	acct_on	75%
3	acct_auto_close_mnt	82%
4	acct_auto_close	82%
5	acct_exit_ns	83%
6	comp_t encode_comp_t	78%
7	comp2_t encode_comp2_t	78%
8	encode_float	82%
9	acct_collect	74%
10	acct_process_in_ns	79%
11	acct_process	83%
12	async_cookie_t lowest_in_progress	79%
13	async_run_entry_fn	75%
14	async_cookie_t __async_schedule	74%
15	async_synchronize_full	92%
16	async_unregister_domain	86%
17	async_synchronize_full_domain	91%
18	async_synchronize_cookie_domain	81%
19	async_synchronize_cookie	90%
20	open_arg	86%
21	open_arg	86%
22	open_arg	86%
23	audit_set_auditable	86%
24	put_tree_ref	79%
25	grow_tree_refs	81%

3.2.2 Maintainability Index for Modules

After calculating the MI for each separate function, wants to calculate the MI for each module which is the average of the MIs for its functions. So that, using the chi square test to test the type of distribution for the MI parameters (LOC, CC, and HV).

3.2.2.1 Chi square for Line of Code

H_0 = each Line of Code have same probability of occurrence

H_1 = each Line of Code does not have same probability of occurrence

Table 3: Chi Square test for LOC

$\boldsymbol{E}$	12.67
Degree of Freedom	65
Calculated $(\boldsymbol{X^2})$	3299

From freedom degree table with the degree of freedom 65 and Confidence level 0.95 got Chi square results is 90.531. By comparing calculated (X^2) resulted that existing in Table 3 and tabular result for Chi square, found that the calculated value is greater than tabular value; so reject H_0 assumption. That mean LOC data not submit uniform distribution. So check normal distribution by dividing the results of LOC into categories and summing the frequency for each category. LOC Subject to the normal distribution, $\mu = 9.78$ and $\sigma = 7.90$ for collecting data.

$\mu_{LOC} - 3\sigma_{LOC} < X_{LOC} < \mu_{LOC} + 3\sigma_{LOC}$ ……..……(7)

Where

μ_{LOC} is mean for LOC.

σ_{LOC} is the standard division (Sigma) for LOC

3.2.2.2 Chi Square for Cyclomatic Complexity

H_0 = each Cyclomatic Complexity have same probability of occurrence

H_1 = each Cyclomatic Complexity does not have same probability of occurrence

Table 4: Chi Square Test for CC

E	32.19
Degree of Freedom	24
Calculated (X^2)	4441

From freedom degree table, under degree of freedom 24 and Confidence level 0.95 got Chi square results is 36.415. By comparing calculated (X^2) resulted that existing in Table 3 and tabular result for Chi square, found that the calculated value is greater than tabular value; so reject H_0 assumption. That means that CC data does not submit uniform distribution. So, check normal distribution by dividing the results of CC into categories and summing frequency for each category. After testing found that CC Subject to the normal distribution with, $\mu = 19.46$ and $\sigma = 13.29$ for the collected data.

$\mu_{cc} - 3\sigma_{cc} < X_{cc} < \mu_{cc} + 3\sigma_{cc}$ ……..……(8)

Where :

μ_{CC} is mean for CC.

σ_{CC} is the standard deviation for CC

3.2.2.3 Chi Square for Halsted Volume

H_0 = each Halsted Volume have same probability of occurrence

H_1 = each Halsted Volume does not have same probability of occurrence

Table 5: Chi Square Test for HV

E	1.31
Degree of Freedom	347
Calculated (X^2)	119.26

From freedom degree table, under degree of freedom 347 and Confidence level 0.95 got chi square results is 124.342. By comparing calculated (X^2) resulted that existing in Table 3 and tabular result for Chi square, found that the calculated value is greater than tabular value; so accept H_0 assumption. That means that HV submit uniform distribution or the probability of occurrence for each one = $1/n$. And the mean value for HV

$$HV = \left(\frac{\sum_{i=1}^{n}(FHV_i)}{n}\right) \text{ ……..……(9)}$$

Where:

FHV is Halsted Volume for functions.

n is number of functions

From above statistical tests for LOC, CC and HV, compute maintainability index for module by the following equation

$$MI = 171 - 5.2\ln[HV] - 0.23[X_{cc}] - 16.2\,ln[X_{LOC}] \text{ ...(10)}$$

3.2.3 Correlation Coefficient Calculation

After calculating the Chi square for each parameter of MI, calculate the Correlation Coefficient for each parameter versus the other parameters.

Determine the correlation between each parameter of the MI with the other parameters and with the MI itself. These are presented in forms of propagation in Figure 2-7 consequently.

3.2.4 Line of Code (LOC) Correlations

Using a simple correlation analysis between two variables, and apply it to LOC vs. CC, LOC vs. HV and LOC vs. MI. Reached, that the correlation between LOC vs. CC is an extreme correlation with value 0.8784. As seen in Figure 2, where X axis represented LOC and Y axis represented CC, dots represented intersection between LOC values and CC values for each function sequentially and line represent correlation coefficient between LOC and CC.

And There is a Very strong correlation extrusive between LOC vs. HV with value 0.9559 but there is a There is a strong inverse correlation between LOC vs. MI with value -0.8570

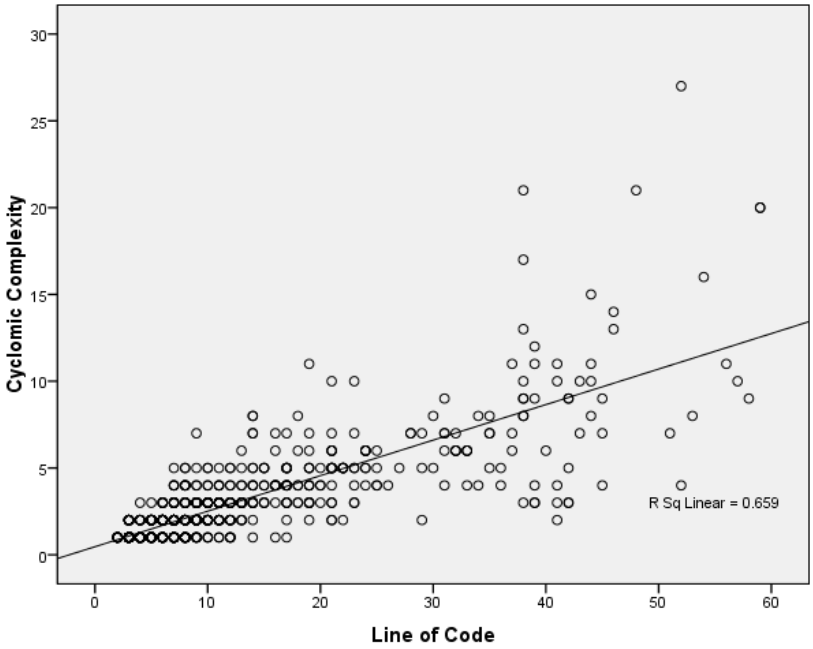

Figure 2: Correlation Coefficient between LOC vs. CC

Also, the correlation between LOC vs. HV is extreme correlation with value 0.9559. As seen in Figure 3, where X axis represented LOC and Y axis represented HV, dots represented intersection between LOC values and HV values for each function sequentially and line represent correlation coefficient between LOC and HV.

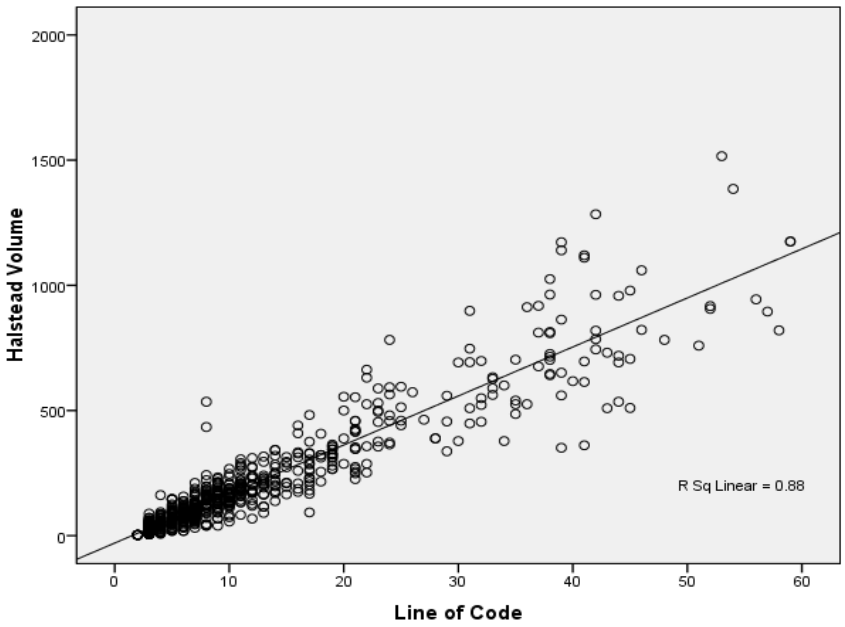

Figure 3: Correlation Coefficient between LOC vs. HV

However, the correlation between LOC vs. MI is inverse correlation with value -0.8570. As seen in Figure 4, where X axis represented LOC and Y axis represented MI, dots represented intersection between LOC values and MI values for each function sequentially and line represent correlation coefficient between LOC and MI.

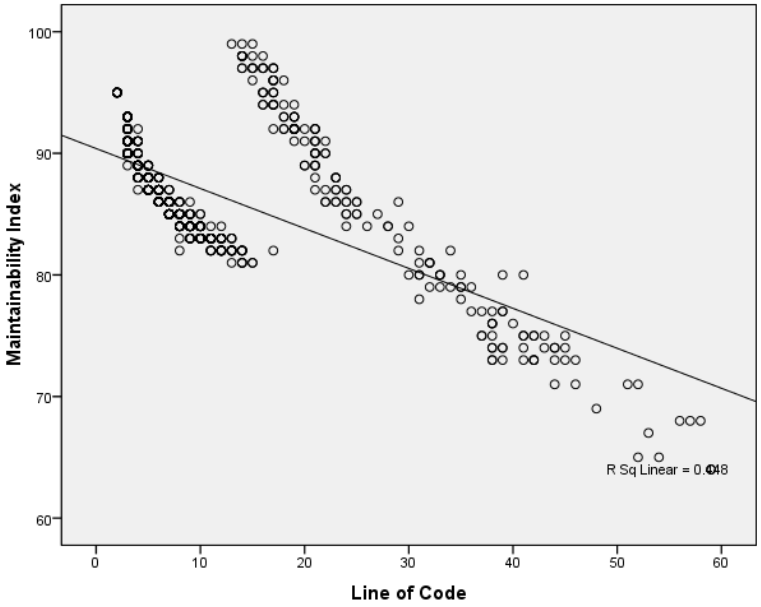

Figure 4: Correlation Coefficient between LOC vs. MI

3.2.5 Cyclomatic Complexity (CC) Correlations

Also Appling simple correlation coefficient for CC vs. HV and CC vs. MI. reaching that the correlation between CC vs. HV is extreme correlation with value 0.8320. As seen in Figure 5, where X axis represented CC and Y axis represented HV, dots represented intersection between CC values and HV values for each function sequentially and line represent correlation coefficient between CC and HV.

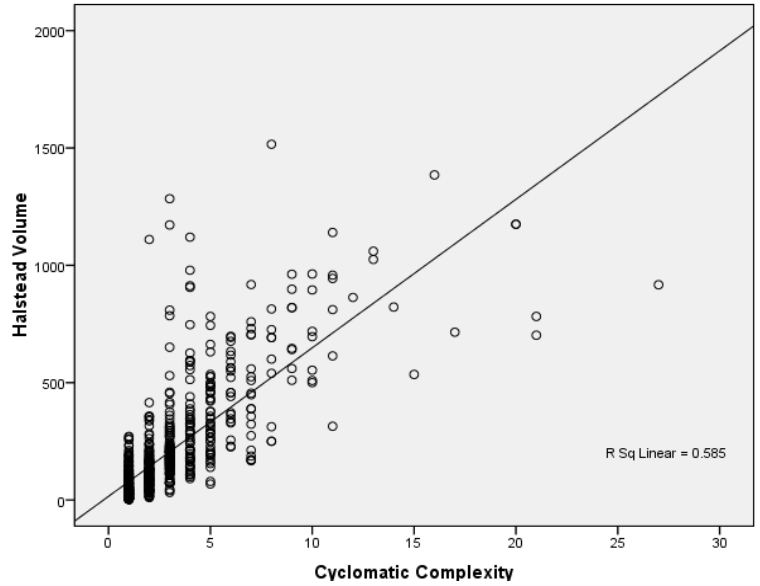

Figure 5: Correlation Coefficient between CC vs. HV

However, the correlation between CC vs. MI is inverse correlation with value -0.7436. As seen in Figure 6, where X axis represented CC and Y axis represented MI, dots represented intersection between CC values and MI values for each function sequentially and line represent correlation coefficient between CC and MI.

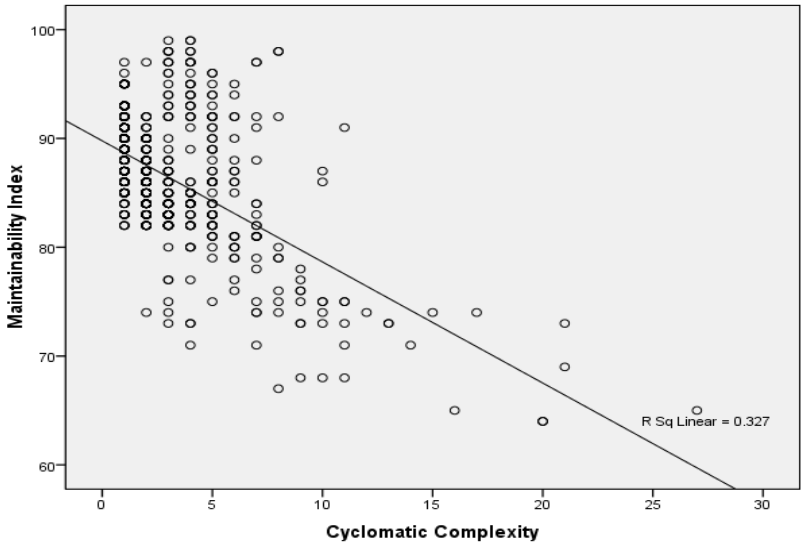

Figure 6: Correlation Coefficient between CC vs. MI

3.2.6 Halstead Volume (HV) Correlations

Finally as seen in the correlation between HV vs. MI is inverse correlation with value -0.8099. as seen in Figure 7 where X axis represented HV and Y axis represented MI, dots represented intersection between HV values and MI values for each function sequentially and line represent correlation coefficient between HV and MI.

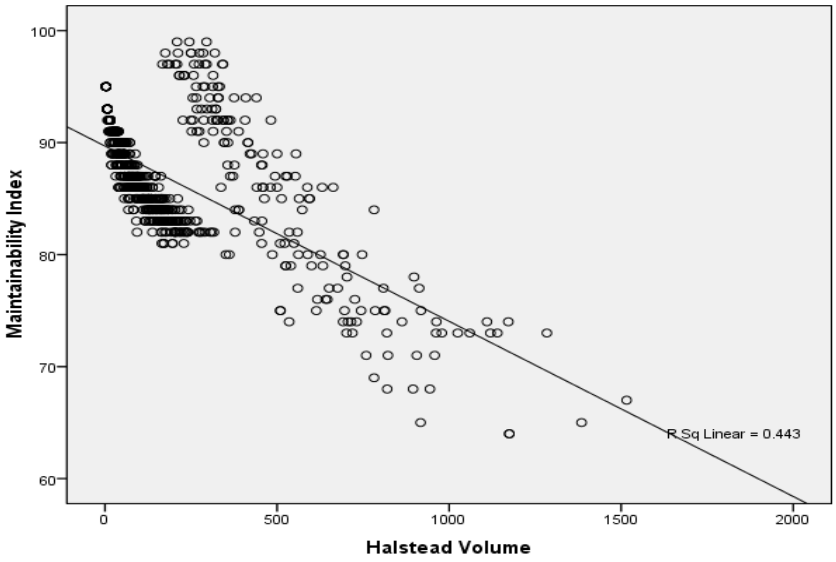

Figure 7: Correlation Coefficient between HV vs. MI

3.3 Analytical Hierarchy Process Calculation

In this phase computing AHP for MI Parameters (LOC, CC and HV). The priority of each component based on correlation coefficient results for each parameter and other one. Assume the maximum correlation coefficient be the main parameter is very strongly more important than another parameter, the middling correlation coefficient be the main strongly more important anther parameter and the minimum correlation coefficient be the main weakly more important anther parameter.

Using AHP methodology and according to correlation coefficient results, where the correlation coefficient between LOC vs. HV is heights result. So assume that the priority for LOC with HV by 7; so the priority for HV with LOC is 1/7, where the correlation coefficient for LOC vs. CC is middling so the priority for LOC with CC is 5; so the priority for CC with LOC priority is 1/5, and where correlation coefficient between CC vs. HV the priority for CC with HV is 3; so the priority for HV with CC is 1/3 as see in Table 6

Table 6 : Priority for each parameter vs. another parameter

	LOC	CC	HV
LOC	1	5	7
CC	$1/5$	1	$1/3$
HV	$1/57$	3	1

Table 7 and Figure 8 shows the LOC is the most important parameter for computing MI with weight 75.5%, the HV is

second important parameter with weight 16.11% and the third important parameter is HV with weight 8.38%.

Table 7: Weight value for each parameter

	AHP Result	Percentage
LOC	0.755	75.5%
CC	0.0838	8.38%
HV	0.1611	16.11%

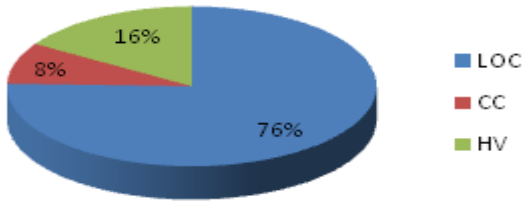

Figure 8: Weight of MI parameters

4. CONCLISION

In this research, in order to calculate the maintainability index of open source code software modules, separating modules to its component functions. Then calculate each function maintainability index using Line of Code, Cyclomatic Complexity and Halstead Volume for each one.

To calculate maintainability index for a module, need to calculate averages of Line of Code, Cyclomatic Complexity and HV for the functions of each module. Using Chi square test proved that Line of Code and Cyclomatic Complexity values are normally distributed. So Line of Code come by $\mu = 9.78$ and $\sigma = 7.90$ so:

$$\mu_{LOC} - 3\sigma_{LOC} < X_{LOC} < \mu_{LOC} + 3\sigma_{LOC}$$

And the mean of Cyclomatic Complexity $\mu = 19.46$ and $\sigma = 13.29$ so:

$$\mu_{cc} - 3\sigma_{cc} < X_{cc} < \mu_{cc} + 3\sigma_{cc}.$$

In addition, by using correlation coefficient model, practically proved that there is a relation between Line of Code, Cyclomatic Complexity, Halstead Volume and Maintainability Index. There is a strong correlation extrusive between Line of Code vs. Cyclomatic Complexity, Cyclomatic Complexity vs. Halstead Volume. Also there is a very strong correlation extrusive between Line of Code vs. Halstead Volume. However, there is a strong inverse correlation between Maintainability Index vs. Line of Code, Maintainability Index vs. Cyclomatic Complexity and Maintainability Index vs. Halstead Volume.

Finally by using Analytical Hierarchy Process, practically proved that the most important parameter which effect on maintainability index is Line of Code with weight value about 76%, the second parameter is Cyclomatic Complexity with weight value about 8% and the last parameter is Halstead Volume with weight value about 16%.

5. REFERENCES

[1] Jeff Tain, 2005, "Software Quality Engeneering", Published by John Wiley & Sons.

[2] Ian Sommerville, 2007, "Software Engineering", 8th, published by arrangement

[3] Preliminary report, 2003, "Software engineering Product Quality," ISO copyright office, Switzerland.

[4] T. J. McCAB, 1976, "A Complexity Measure," IEEE, TRANSACTIONS On Software ENGINEERING, Vols. SE-2, No4.

[5] R. Land, 2003, "Measurements of Software Maintainability," Mälardalen University, Department of Computer Engineering.

[6] Anita Ganpati, Dr. Arvind Kalia, Dr. Hardeep Singh, October 2012, "A Comparative Study of Maintainability Index of Open Source Software" IJETAE, Volume 2, Number 10, Pages 228 – 230

[7] Ilja Heritage, Tobias Kuipers, Joost Visser, 2007, "A Practical Model for Measuring Maintainability," preliminary report, Netherlands.

[8] Don Coleman and Dan Ash, 2009, "Using Metrics to Evaluate software System Maintainability," IEEE.

[9] S. D. Conte, Herbert E. Dunsmore,. V. Y. Shen, 1981, "Software Science Revisited: A Critical Analysis of the Theory and Its Empirical Support", Purdue University-Purdue e-Pubs, Department of computer science

[10] Dimitris Stavrinoudis , Greece Michalis Xenos , Greece Dimitris Christodoulakis, 1999, "Relation between software metrics and maintainability", FESMA99 International Conference, Federation of European Software Measurement Associations, Amsterdam, The Netherlands, pp. 465-476.

[11] Meine J.P, Miguel A. Revilla, 2007, "Correlation between Internal software Metrics and Software Dependability in a Large Population of Small C/C++ Programs", 18th IEEE, Pages 203 -208

[12] Mukti Chauhan, Monika Sharma, june 2013, "Predicting Maintainability Of Open Source Softwares: An Empirical Approach", IJERT, Volume 2, Number 6, Pages 3333- 3336

[13] Marie Diener-West, 2008,"Use of the Chi-Square Statistic", University and Marie Diener-West.

[14] Michael Lavine, 2009, " Introduction to Statistical Thought", Orange Grove Texts Plus

[15] Sarah Boslaugh, 2012, " Statistics in a Nutshell, 2nd Edition", Publisher: O'Reilly Media, Inc.

[16] Evangelos Triantaphyllou , Stuart H. Mann, 1995, "Using the Analytic Hierarhy Process for decision making in engeneering applications some challenges" Inter'l Journal of Industrial Engineering: Applications and Practice, Vol. 2, No. 1, pp. 35-44.

[17] Doraid Dalalah, Faris AL-Oqla, Mohammed Hayajneh, November 2010, "Application of the Analytic Hierarchy Process (AHP) in Multi-Criteria Analysis of the Selection of Cranes",JJMIE, Volume 4, Number 5, Pages 567 – 578

[18] Halstead Metrics," Verify Soft Technology, [Online]. Available: http://www.verifysoft.com/en_halstead_metrics.html. [Accessed Feb 2014].