Correlation Analysis of Maintainability Index of Server-Side Script: PHP

Cho Thet Mon, Khin Mar Myo
University of Computer Studies, Mandalay
chothetmonucsm@gmail.com,kmmyo.ag@gmail.com

Abstract

The maintainability of any software system is quantified in terms of Maintainability Index (MI). Many research papers used MI as Maintainability indicator to validate and predict the maintainability of their proposed metrics. In this study the maintainability change of the PHP language was empirically investigated. The research performed on 210 PHP source codes from different domains to compute a maintainability index of each file. The correlations between each parameter for Maintainability Index (MI) with the other parameters and with the MI itself are calculated. The parameters of MI are Line of Code (LOC), Cyclomatic Complexity (CC), and Halstead Volume (HV). The relationship between maintainability index and metrics taken for the study was identified on the basis of the Pearson correlation analysis. From the results it can be depicted that the Line of Code (LOC), Cyclomatic Complexity (CC), and Halstead Volume (HV) are strongly inversely related to the maintainability index of PHP.

1. Introduction

Measuring software maintainability early in the development life cycle, especially at the design phase, may help designers to incorporate required enhancement and corrections for improving maintainability of the final software [5]. In order to effectively manage the cost of the software development it is important to forecast software's maintainability and identify maintainability predictors which have an impact on the software maintenance activity. Many software metrics have been proposed as indicators for software product quality in particular, Oman et al. proposed the Maintainability Index (MI) [1 and 4]. MI is a composite metric that incorporates a number of traditional source code metrics into a single number that indicates relative maintainability. The MI is comprised of weighted Halstead metrics (effort or volume) HV, McCabe's cyclomatic complexity (CC), Lines of codes (LOC). This Maintainability Index (MI) has evolved into numerous variants and has been successfully applied to a number of industrial strength software systems. After nearly a decade of use, MI continues to provide valuable insight into software maintainability issues.

2. Literature Review

Several maintainability models or methodologies were proposed to help the designers in calculating the maintainability of software so as to develop better and improved software systems. Many organizations assess the maintainability of software systems before they are deployed. Object-oriented design has been shown to be a useful technique to develop and deliver quality software. Object-oriented metrics can be used to assess the maintainability of a software system. Various software metrics and models have been developed and described. Wide range of maintainability prediction models has been proposed in the literature within last two decades. Some of the models are predicting maintainability using the metrics from coding as well as the design phase, while some are focusing only on design level metrics.

Oman and Hagemeister [4] proposed a software maintainability hierarchy, in terms of some maintainability indicators and as per the hierarchy, Halstead Complexity and Cyclomatic Complexity are the indicators of maintainability.

Anita Ganpati et al. [2] proposed the maintainability index observed over fifty successive versions, applied on Apache, Mozilla Firefox, and MySQL and FileZilla software. The MI in terms of software metrics namely Lines of Code (LOC), Cyclomatic Complexity (CC), and Halstead Volume (V) was computed for all the fifty successive versions of four open source software (OSS). The software metrics were calculated using Resource Standard Metrics (RSM) tool and Crystal Flow tool. It was observed from the results that the MI value was the highest in case of Mozilla Firefox and was the lowest in the case of Apache OSS.

Nahlah M.A.M.Najm [3] provided some insight to the practical implementation of MI with a new, simple, and effective method in comparison with the traditional method. This paper presented a new method to find MI with respect to LOC only. To validate the method, Measuring Maintainability Index Software (MMIS) is developed, that first finds MI with respect to (LOC, CC, and HV); secondly finds MI with respect to LOC only. The findings proved that the new method was easy to understand, fast to count, and independent on the program language.

The main purpose of this work is that maintainability of PHP is calculated. Three internal software metrics LOC, CC and HV are used to calculate the maintainability index. Also, Pearson correlation coefficient is calculated for each parameter of Maintainability Index (MI) with the other parameters and with the MI itself.

3. Maintainability Index (MI)

Maintenance is defined by the IEEE as “the process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment” [1]. There have been several attempts to quantify the maintainability of a software system. The most widely used software metric which quantifies the maintainability is known as Maintainability Index (MI).

Maintainability Index is software metric which measures how maintainable (easy to support and change) the source code is. The maintainability index is calculated as a factored formula consisting of Lines of Code (LOC), Cyclomatic Complexity (CC) and Halstead Volume (HV).

According to Coleman, a MI value above 85 indicates that the software is highly maintainable, a value between 85 and 65 suggests moderate maintainability, and a value below 65 indicates that the system is difficult to maintain [1]. First we need to measure the following metrics from the source code:

-HV = Halstead Volume

-CC = Cyclomatic Complexity

-LOC = count of source Lines of Code

From these measurements the MI can be calculated [1].The original polynomial equations defining MI are as follows:

$$MI = 171 - 5.2 * \ln(HV) - 0.23 * (CC) - 16.2 * \ln(LOC) \quad ....(1)$$

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).

HV=N log2 n

Where:

-n1= the number of distinct operators

-n2= the number of distinct operands

-N1= the total number of operators

-N2= the total number of operands

-Program vocabulary: n =n1+n2

-Program length: N= N1+N2

CC = no. of condition statements + no. of loops statements + 1

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

4. Research Methodology

PHP is a general-purpose server-side scripting language designed for Web development to produce dynamic Web pages. Not only the separated PHP codes but also the whole project can be tested to measure and analyze MI of PHP. In this study, a number of 210 PHP source codes were used as the data sources that are collected from different domains. These codes are fed as inputs to proposed measuring tool to assess maintainability. The various software metrics namely LOC, CC, Halstead Volume and MI required for studying the maintainability index change were calculated. Also, the Pearson correlation analysis is used to test the distribution for each of the MI parameters.

5. Experimental Results

In this analysis, the different features of 210 PHP codes from different sources mainly from www.php.net web site are tested and they are divided into 3 groups according to their input features that are ranged such that ID-1 to ID-45 are web application PHP, ID-46 to ID-90 are procedural PHP and ID-91 to ID-210 are object-oriented PHP. The results for the maintainability index of 210 PHP codes are depicted in Figure 1. According to the quantitative analysis results, most of the MI values were increased for web application and procedural PHP codes. That is because most of web application programs are especially written for user interface forms that contain less complexity values. And procedural PHP files are the basic PHP programming files that are easier to understand that lead to maintainable codes and vice versa they did not contain complex features.

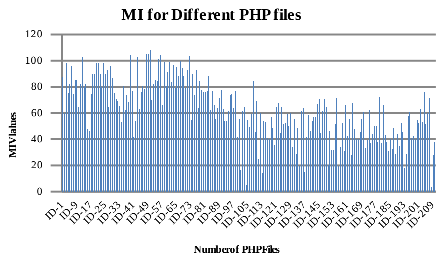
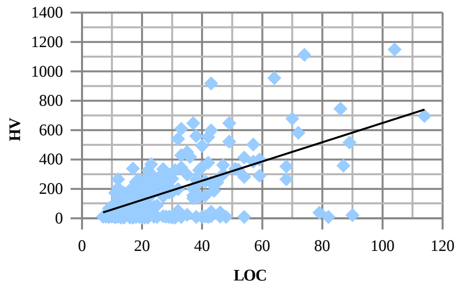


Figure 1.Maintainability Index for Different PHP Files

6. Correlation Analysis

There are a lot of studies done on measuring software metrics and analyzing the correlations between them to determine the way software characteristics are influencing each other. To verify this, the relation between them is studied by applying correlation indicators, PEARSON coefficient being one of them [6]. We analyzed the collected dataset by calculating correlation coefficient for each pair of metrics. The correlation coefficient is a numerical value between -1 and 1 that expresses the strength of the linear relationship between two variables. When r is closer to 1 it indicates a strong positive relationship. A value of 0 indicated that there is no relationship. The values close to -1 signal a strong negative relationship between the two variables. We explore 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 to Figure 7 consequently.



To determine the significant of sample correlation, we need to use a critical value table to examine weak or strong correlation between two variables [7]. The obtained correlation coefficients that are significant are set in boldface. The chosen significant level is p-value =0.05. This means that for the corresponding pairs of metrics there exists a correlation at the 95% confidence level.

6.1. Line of Code (LOC) Correlations

Using a simple correlation analysis between two variables, the relations among LOC vs. CC, LOC vs. HV and LOC vs. MI are explored.

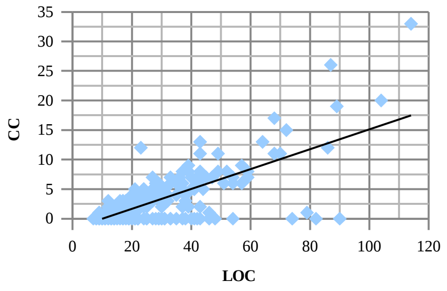


Figure 2. Correlation Coefficient between LOC and CC

The obtained relation stated that the correlation between LOC vs. CC is an extreme correlation with value 0.703119.

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.

There is a significant correlation between LOC vs. HV with value 0.599221.

Figure 3. Correlation Coefficient between LOC and HV

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.

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.

The negative correlation between LOC and MI with value -0.80606 indicated that as the size of program increased the MI seem to decrease.

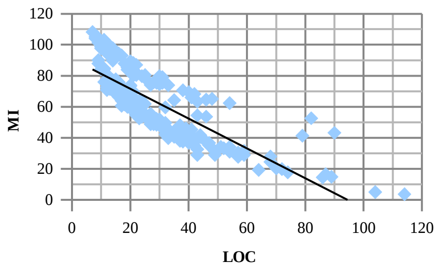


Figure 4. Correlation Coefficient between LOC and MI

6.2. Cyclomatic Complexity (CC) Correlations

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.

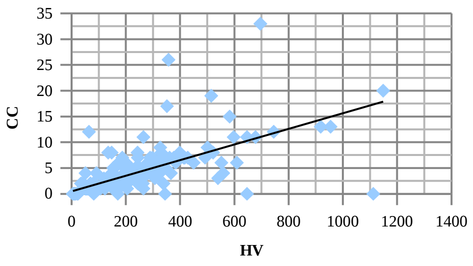


Figure 5. Correlation Coefficient between HV and CC

As seen in Figure 5, where X axis represented HV and Y axis represented CC, dots represented intersection between CC values and HV values for each function sequentially and line represent correlation coefficient between CC and HV.

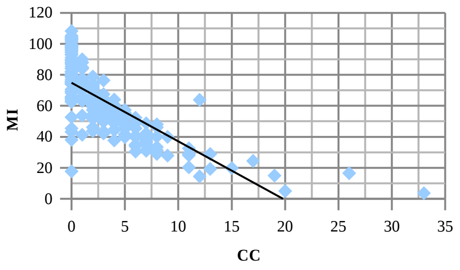


Figure 6. Correlation Coefficient between CC and MI

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.

There was a very significant negative correlation between CC and MI with value -0.756. The more complex a piece of software, the more effort is required to maintain it. The higher the software complexity, the more difficult it is to understand its source code for maintenance and evaluation purposes. The observed relation indicated that complexity metric has been demonstrated to have a strong negative correlation with MI.

6.3 Halstead Volume (HV) Correlations

Finally as seen in the correlation between HV vs. MI is inverse correlation with value -0.812.

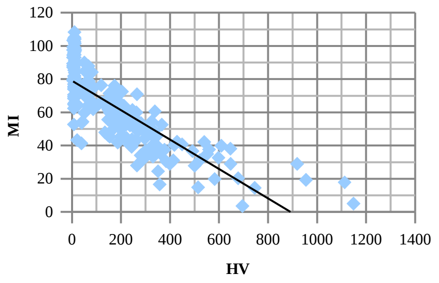


Figure 7. Correlation Coefficient between HV and MI

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.

7. Conclusion

Using the MI to assess source code and thereby identify and quantify maintainability is an effective approach. The MI provides an excellent guide to direct human investigation. This paper provides some insight to the practical implementation of MI with different domains. It is observed that there is a decrease in the maintainability index of object-oriented PHP. Moreover, the correlation analysis results have shown that the software metrics namely CC, LOC, Halstead volume are inversely related to the maintainability of PHP software. In particular, depending on the results, the HV and LOC can be considered as the most important factor for controlling the maintainability of the PHP.

References

[1] D. M. Coleman, D. Ash, B. Lowther, and P. W. Oman, “*Using Metrics to Evaluate Software System Maintainabilit*”, IEEE Computer, vol. 27, no. 8, pp. 44–49, 1994.

[2] A. Ganpati, Dr. A. Kalia, Dr.H. Singh, October 2012, “*A Comparative Study of Maintainability Index of Open Source Software*” IJETAE, Volume 2, Number 10, Pages 228 – 230

[3] M.A.M. Nahlah, “*Measuring Maintainability Index of a Software Depending on Line of Code Only*”, IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 2, Ver. VII (Mar-Apr. 2014), PP 64-69.

[4] P. Oman and J. Hagemeister, “*Metrics for assessing a software system's maintainability*,” Proceedings of the Conference on Software Maintenance, IEEE Computer Society Press, pages 337–344, 1992.

[5] S. W. A. Rizvi and R. A. Khan, “*Maintainability Estimation Model for Object-Oriented Software in Design Phase (MEMOOD)*”, JOURNAL OF COMPUTING, VOLUME 2, ISSUE 4, APRIL 2010, ISSN 2151-9617.

[6]http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient.

[7]http://faculty.fortlewis.edu/CHEW_B/Documents/Table of critical values for Pearson correlation.htm