Maintainability Index over Multiple Releases: A Case Study PHP Open Source Software

Anita Ganpati
Dept. of Computer Science
Himachal Pradesh University
Shimla, India

Dr. Arvind Kalia
Dept. of Computer Science
Himachal Pradesh University
Shimla, India

Dr. Hardeep Singh
Dept. of Computer Science
G N D University
Amritsar, India

Abstract

Software maintainability is the ease by which a software system can be modified. Maintainability of the software is an important software attribute which represents the majority of the costs in software development life-cycle. 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. The maintainability of any software system is quantified in terms of Maintainability Index (MI). In this study the maintainability change in terms of MI of the PHP Open Source Software (OSS) of fifty releases over a period of seven years was empirically investigated. Also the relationship between various software metrics namely Lines of Code (LOC), Cyclomatic Complexity (CC), Halstead Volume and MI for fifty versions of the PHP was investigated. The software metrics were calculated using Resource Standard Metrics (RSM) tool and Crystal Flow tool. The results indicated that there is a decrease in the MI of the PHP over its successive releases. 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 size and complexity metrics are strongly inversely related to the maintainability index of PHP.

1. Introduction

Change to software is inevitable. All software systems must change with time to meet the growing needs of its users [5]. Software maintenance is the process of enhancing and optimizing existing software [16]. Therefore, it is necessary for the software development houses, software developers and software development teams to perform software maintenance in such a way that the difficulties arising from changes are reduced. In Open Source Software (OSS) development scenario, software is available on the internet and it allows developers around the world to contribute to the code, facilitate addition of new functionalities, improvement of exiting software version and submitting bug fixes to the current release. Developing an OSS system implies a series of frequent maintenance efforts for debugging, existing functionality and adding new ones to the system. In the OSS, the development as well as maintenance of the software is decentralized due to which the maintainability is core issue in OSS development. Software maintainability is an attribute that characterizes the ease with which the existing source code can be modified to provide new or changed functionality, correct faults, improve performance, or adapt to a changed environment [14]. The software products should remain maintainable otherwise they have to be reengineered [13]. 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) [15].

The various software design and code metrics can be useful for predicting maintainability of OSS [10]. Any software system's maintainability is quantified in terms of MI. MI is a combination of software metrics namely McCabe's Cyclomatic Complexity (CC), Halstead's Volume (V) and Lines of Code (LOC) that affect maintainability of the software [8]. More precisely, MI is defined as follows:

$$MI = 171 - 5.2 * \ln(aveV) - 0.23 * aveV(g) - 16.2 * \ln(aveLOC)$$

where

aveV= average Halstead Volume

aveV(g) = average Cyclomatic Complexity per module

aveLOC = average Lines of Code per module

MI is an index which has values ranging from 0 to 100, predicting the ease of the maintenance of the code. A MI value of 100 indicates that the software or code has very good maintainability whereas a 0 value indicates that the code is very difficult to maintain if not impossible [7]. According to Oman et al. the code with MI value above 85 is highly maintainable. The MI value of the code between 65 and 85 is moderately maintainable. The code with MI value below 65 is difficult to maintain [11]. In order to assess the association between

maintainability index and the different software metrics, an empirical study is conducted. In other words, it is intended to observe the statistical nature and significance of a relationship between the software metrics (size & complexity) and maintainability index of the software.

2. Literature Review

Coleman et al. proposed Maintainability Index as the most common approach for maintainability estimation [4].
Hayes et al. [6] derived a model for estimating adaptive maintenance effort. It was concluded that the number of LOC changed and the number of operators changed are strongly correlated with the maintenance effort.
Aggarwal et al. [1] concluded that three main factors affecting software maintainability are readability of source code, documentation quality and understandability of software.
Banker et al. [3] analyzed the relationships between software complexity and maintenance and recommended that complexity has a significant impact on maintenance costs.
Oman et al. [12] 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.
Ash et al. analyzed the maintainability index for two software products using the four-metric model over several releases [2].
Denis et al. investigated the relationships between software maintainability and other internal software quality attributes. The source code characteristics of five Java-based open-source software products were analyzed. It was concluded that the number of data variables declared and the McClure metric for decisional complexity have the strongest correlations with maintainability [9].

3. Objective of the Study

The specific objectives of this study are:
1) To investigate the maintainability index change for fifty versions of the PHP software over the period of seven years.
2) To identify the relationship between various software metrics (LOC, CC, Halstead Volume and Maintainability Index) of fifty versions of the PHP.

4. Research Methodology

The source code of open source software PHP was used in this study as the data source. PHP is a general-purpose server-side scripting language designed for Web development to produce dynamic Web pages. The various software metrics namely LOC, CC, Halstead Volume and MI of fifty versions of the open source software PHP required for studying the maintainability index change were calculated using software tools namely Resource Standard Metrics (RSM) and Crystal Flow. The relationship between maintainability index and various metrics under study were identified on the basis of the Pearson correlation analysis. The correlation values were computed by using a public social private partnership (PSPP) tool. PSPP is a free software application for analysis of sampled data. It is particularly suited to the analysis and manipulation of very large data sets. In addition to statistical hypothesis tests such as Pearson correlation, t-tests, analysis of variance and non-parametric tests PSPP can also perform linear regression. Also it is a very powerful tool for recoding and sorting of data and for calculating metrics such as skewness and kurtosis[17].

5. Analysis

The Maintainability Index of the PHP software was observed over a period of seven years. The results for the maintainability index for fifty versions of PHP are graphically depicted in the Figure 1. It is clear from the values of the maintainability index that the there is a decrease in the maintainability index of PHP in its successive versions. The maintainability index for the version 4.3.9 which was released in the year 2004 is 82.3 however the latest PHP version 5.3.7 released in the year 2011 has the MI value 75.9. The decrease in the maintainability index is an indicator that the maintenance of the PHP has increased over the successive releases.

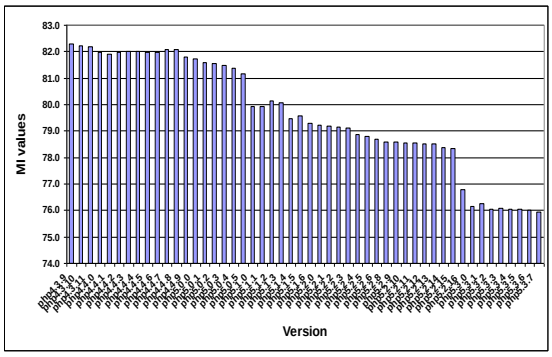

Figure 1. Graph between Maintainability Index (MI) values and different Versions

Also the correlation values between for the LOC, CC, Average Halstead Volume and MI were computed to identify the relationship between the size, complexity metrics and maintainability index.

Table 1. Correlation between LOC, CC, Halstead Volume and MI

Software Metrics	Maintainability Index (MI)
Avg. LOC	-0.94
Avg. CC	-0.93

Avg. Halstead Volume	-0.89

It is clear from the correlation results in the Table 1 that there is a strong negative correlation between the Average CC and MI, Average LOC and MI, Average Halstead Volume and MI. It indicates that the increase in CC, LOC and Halstead Volume will decrease the maintainability of the software. The MI for any software system quantifies the maintainability of the software. Therefore from the results it can be deduced that various software metrics namely CC, LOC and Halstead Volume can be considered as the controlling factors for the maintainability of software.

6. Conclusion & Future Scope

The maintainability index of the PHP was analysed over a period of seven years. It was observed that there is a decrease in the maintainability index of the PHP. So it can be concluded that the maintenance of PHP has increased over the successive releases. Moreover the correlation analysis results have shown that the software metrics namely CC, LOC, Halstead volume are strongly inversely related to the maintainability of PHP software. In particular, the CC and LOC can be considered as the most important factor for controlling the maintainability of the PHP. In future, more empirical studies with more number of software over a long period in term of number of years can be carried out to strengthen the viewpoint.

7. References

[1] K.K. Aggarwal, Y. Singh and J.K. Chhabra, " An Integrated Measure of Software Maintainability", *Annual Proceedings on Reliability and Maintainability Symposium*, IEEE Computer Society Press, pages 235–241, 2002.

[2] D. Ash, J. Alderete, L. Yao, P.W. Oman and B. Lowther, "Using Software Maintainability Models to Track Code Health",*Proceedings of the International Conference on Software Maintenance*, IEEE Computer Society Press; pages 154–160, 1994.

[3] R. Banker, S. Datar, C. Kemerer and D. Zweig, "Software Complexity and Maintenance Costs" , *Communications of the ACM*; 36 (11), pages 81–94, 1993.

[4] D. Coleman, D. Ash, B. Lowther and P. Oman, " Using Metrics to Evaluate Software System Maintainability", *IEEE Computer*; 27(8), pages 44–49, 1994.

[5] Penny Grubb, Armstrong A. Takang, "*Software Maintenance: Concepts and Practice*", World Scientific Publishing Co. Pvt. Ltd., Singapore, 2003.

[6] J. Hayes, S. Patel, L. Zhao, "A Metrics-Based Software Maintenance Effort Model". *Proceedings 8th European Conference on Software Maintenance and Reengineering*, IEEE Computer Society Press, pages 254–260, Los Alamitos ,Calfornia, 2004.

[7] Don Jones, "*The Definitive Guide to Building Code Quality",* Real Time Publishers.

[8] R. A. Khan, K. Mustafa and S.I. Ahson, "*Software Quality: Concepts and Practices",* Narosa Publishing House.

[9] Denis Kozlov, Jussi Koskinen, Markku Sakkinen and Jouni Markkula, " Assessing Maintainability Change Over Multiple Software Releases", *Journal of Software Maintenance and Evolution: Research and Practice*, 20 (1), pages 31–58, 2008.

[10] Subhas Chandra Misra, "Modeling Design/Coding Factors That Drive Maintainability of Software Systems", *Software Quality Journal*, 13, pages 297-320, 2005.

[11] Paul W. Oman and Shari Lawrence Pfeeleger, " *Applying Software Metrics",* Johnn Wiley and Sons Publishers.

[12] 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.

[13] R. S. Pressman, "*Software Engineering: A Practitioner's Approach",* 5th ed., McGraw-Hill, 2001.

[14] I. Sommerville, "*Software Engineering",* 7th ed., Pearson Addison Wesley, 2004.

[15] E. VanDoren, "Maintainability Index Technique for Measuring Program Maintainability", *Technical Report*, Software Engineering Institute, March 2002.

[16] Liguo Yu, "Indirectly Predicting the Maintenance Effort of Open-Source Software", *Journal of Software Maintenance And Evolution: Research And Practice*, 18, pages 311–332, 2006.

[17] Arthur Griffith. *SPSS For Dummies*. 2nd ed., Wiley Publication.