

Performance of MI Tools in Perspective of Open Source Software

Muhammad Imran Sarwar, Wasif Tanveer, and Imran Sarwar

Al-Khwarizmi Institute of Computer Science

University of Engineering and Technology

Lahore, Pakistan

{[mimran.sarwar](mailto:mimran.sarwar@kics.edu.pk),[wasif.tanveer](mailto:wasif.tanveer@kics.edu.pk),[imran.sarwar](mailto:imran.sarwar@kics.edu.pk)}@kics.edu.pk

Abstract—There are numerous ways to determine software maintainability but none of them has been able to become state of the art technique for calculating software maintainability as much as the Maintainability Index (MI). Software maintainability has always been a serious concern in holding all the aces of any software product. Different tools and techniques have been introduced to calculate Maintainability Index of the software system, and there are many MI calculating tools available for calculating software system’s maintainability but the availability of these tools doesn’t mean that it can solve the problem in achieving the software maintainability. Who will look for the flaws in those MI tools? Who will standardize them? This is the emerging issue that is being raised in the current paper. For this purpose we have analyzed behaviors of various open source software products and trailed them on different MI tools which are best available tools for calculating MI. The results highlight the defused behavior of these MI tools and urges for the standardization of calculation mechanism of MI which could lead the software industry to a better and maintainable software systems.

Keywords-component; Maintainability Index, Software maintainability tools, Quality metrics, Software Complexity Measurements

I. INTRODUCTION

Software Maintainability is one of the critical issues that is being faced by the software industry and its decisive importance is elevated with the concept of “Open Source Software Community”. As software system cannot sustain in open source community if the software is not maintainable. The open source software should be more adaptable to the change as compared with the closed source, as open source software is present in a distributed environment [12]. Different metrics have been used to define software maintainability but one of them is used with great acceptance in software industry namely “Maintainability Index (MI)”. To calculate the MI of the system, it is really necessary to identify the right tools that calculate the MI in a very good manner, for this purpose we have conducted a research on studying the results of various MI tools on a number of open source software systems that are vastly used and said to be maintainable.

In our previous study [1] we have shown the effects of different parameters of various open and closed source MI tools, and depending upon those results we will, in this paper, show the effect of those parameters on calculation of MI using different tools on different open source projects, so that we can

judge which tool gives us a better idea towards a maintainable software system. Furthermore this study is an attempt to identify the weaknesses of different MI tools and an effort to rectify those limitations.

While talking about open-source community the word itself comes with the great and colossal impression of maintainable software. That is the reason, why we have chosen some well known open source software and tested them upon different MI calculating tools to see whether these tools call the selected open source software products “Maintainable” in their way of calculating maintainability; it is in this way we might be able to see some revealing facts about the tools that are used to calculate “Maintainability Index” of the software systems, which is a worthy symbol of calculating software’s maintainability.

Previously, we have discussed the tools which are actively involved in claiming that they are the tools that could determine software system’s maintainability in a better way and are having a vast industrial acceptance. But in this current work we have examined the behavior and the results generated by those tools; to give a better and clear picture that tells us the story that there is a dire need of improving the tools and to standardize the way Maintainability Index (MI) is being calculated.

The research paper shows that only the availability of tools is not sufficient to calculate the maintainability of the software systems, but there is a need to specify and standardize a way that the results generated by the tools should be unified, acceptable and comparable. In the current scenario we will see that all tools calculate the “Maintainability Index”. One software system is good according to one tool but it is said to be un-maintainable by the other one. The tools use the same parameters in a way or the other but they have not a standardize way of calculating them. The main emphasis of this research paper is to put the attention towards defining some standard way of calculating the values for the used parameters.

The structure of the paper is as follows: Section II consists of the brief introduction of MI and the previous work done at KICS [8]. Section III discusses and reveals the current problem with the tools available for calculating MI. The results have been shown in the form of graphs and their description is given after each graph in order to evidently describe and understand the problem. In the end we concluded our findings

and gave recommendations on how to improve and standardize the calculation mechanism for MI tools.

Before dilating upon the main topic it appears quite relevant and even desirable to describe some terms related to our discussion that will help us comprehend the concept of software maintainability.

A. Maintainability

Maintainability is the ease with which a software system or component can be modified to correct faults, improve performance, or other attributes, or adapt to a changed environment [4].

B. What's MI

MI is a composite metric derived from the source code to depict the maintainability of the underlying software system. It is based on Halstead Volume (HV) [5], Cyclomatic Complexity (CC) [6], and average number of lines of code (LOC). It also includes an optional measure of percentage of COMment lines per Module (COM). Therefore, there are two formulations to calculate MI, one without comments and the other with comments measurement, referred as 3-Metric and 4-Metric equations respectively. The formula to calculate MI without measuring comments is [3]:

$$3\text{-Metric: } MI = 171 - 5.2 * \ln(\text{aveV}) - 0.23 * \text{aveV}(g') - 16.2 * \ln(\text{aveLOC}) \quad (1)$$

and the formula which includes comments measurement is [3]:

$$4\text{-Metric: } MI = 171 - 5.2 * \ln(\text{aveV}) - 0.23 * \text{aveV}(g') - 16.2 * \ln(\text{aveLOC}) + 50 * \sin(\text{sqrt}(2.4 * \text{perCM})) \quad (2)$$

where

aveV = average Halstead Volume V per module

aveV(g') = average extended cyclomatic complexity per module

aveLOC = the average count of lines of code (LOC) per module

perCM = average percent of lines of comments per module

Halstead Volume is also a composite metric based on the number of distinct operators and operands in the source code.

Halstead Volume is a difficult measure to compute as there is no formal definition of what constitutes an Operator or an Operand in a language such as Java or C# [7].

Cyclomatic Complexity is the number of linearly independent paths through a program.

Lines of code is a software metric used to measure the size of the source code. It typically measures the effort required and the productivity.

Comments per module are the number of comment lines in the source code of a module.

Previously maintainability was not an issue in the software industry, since the focus was current scenarios and current software conditions. However, people now tend to develop

more product oriented software rather than solution oriented, and while considering products it is desirable that they should be maintainable and extensible. For this reason different type of tools and techniques got introduced which helped in determining how much maintainable a software system or product was. This led to the concept of MI (Maintainability Index), which was firstly introduced by Oman et al, of University of Idaho, as described in [2] and the Software Engineering Institute listed structured information about it on its website [3].

MI proved to be very effective in improving software system's maintainability. MI enables a system to be maintenance and cost effective during the software development life cycle [9,10,11]. Moreover, it can be used to analyze and evaluate different systems by comparing their MI values. High risk modules of the source code can be identified with the use of MI. It gives an excellent insight into the source code of a system for direct manual analysis to highlight areas of the code which require human attention.

While applying 4-Metric MI equation (see equation 2) Hewlett Packard (HP) [9] found that software with MI greater than 85 are highly maintainable, while software having MI less than 65, are difficult to maintain. Software having MI values between 65 and 85 have moderate maintainability. So it suggests that greater the MI value, better the chances that the system is maintainable.

II. PREVIOUS WORK/BACKGROUND

As the current software industry has some set standards towards software maintainability, we may have to have a check upon those techniques and tools that they calculate the maintainability in the right way. In our previous work we had defined a set of test cases that will test the effectiveness of the tools [1].

In coming section, we have selected a number of open source software products and calculated their MI values using the selected MI tools and then compared the results generated by these tools to check their performance.

III. MI TOOLS UNCOVERED

MI is useful software metric but from the results generated in this study, this is shown that MI is not calculated through a standard mechanism. This led us to further evaluate various MI tools in detail and their role in improving software system's maintainability. For this reason we had selected various open source software products to analyze/compare their maintainability using MI tools in hand. The list of selected MI tools and the open source software systems is given in Table 1.

TABLE 1. DIFFERENT MI TOOLS AND SELETED OPEN SOURCE SOFTWARE

MI Calculating Tools	Selected Open Source Software
Understand 4 JAVA	Apache ANT
Analyst4J	JUnit
JHAWK	JDOM
	Quartz

TABLE 2. MI VALUES CALCULATED USING DIFFERENT MI TOOLS

Tools Software	Understand 4 JAVA	Analyst4J	JHAWK
Ant	146.17	112.45	132.56
JUnit	69.77	103.34	176.82
JDOM	71.86	102.55	140.21
Quartz	108.84	106.00	114.84

The Table 2 shows the prediction of maintainability of selected open-source software. It is revealed that even the same software is having different MI values if it is evaluated through various MI tools. One software system is maintainable according to one tool and is shown as less or un-maintainable while reviewed with the other MI tool.

The above mentioned table is given in the form of graphs to have a clear picture of them.

When Apache ANT is tested through these selected MI tools the above results are generated which show that according to Understand takes it as more maintainable software but on the contrary Analyst4J does not agree with it and JHawk has a slightly varied opinion than other two tools. The same activity has been performed on selected software systems i.e. JUnit, Quartz, and JDOM to observe the same non-unified behavior of the MI tools.

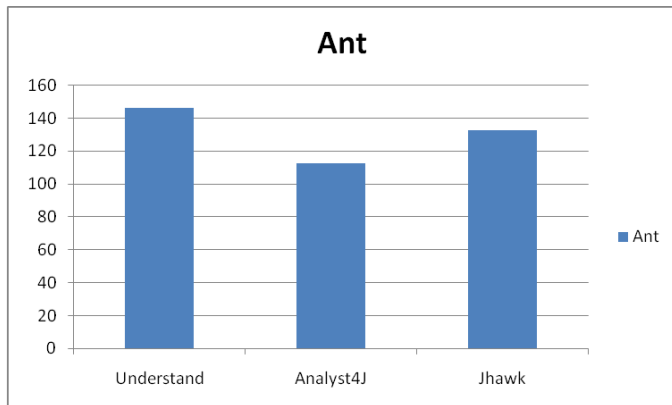


Figure 1. Varied MI Values of ANT

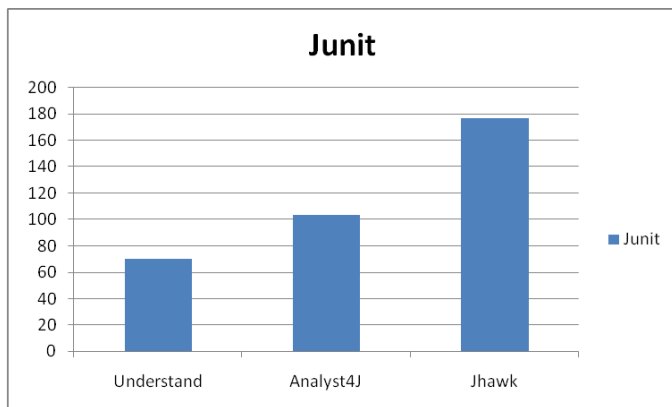


Figure 2. Varied MI Values of JUnit

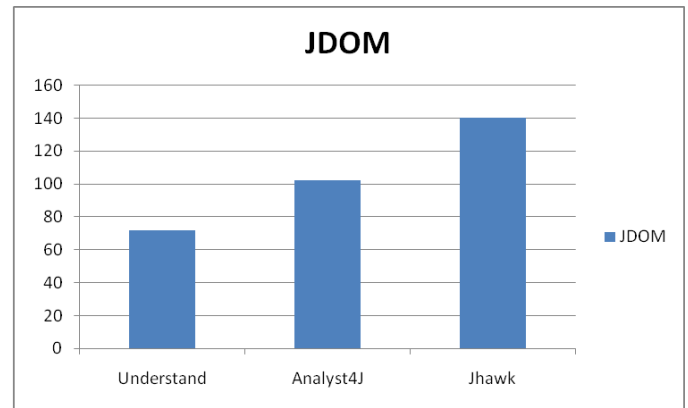


Figure 3. Varied MI Values of JDOM

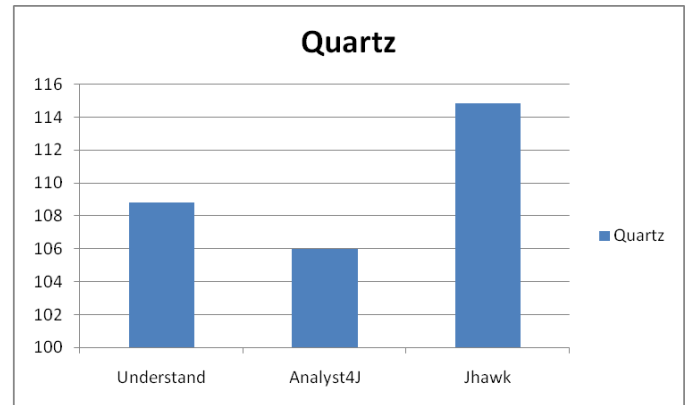


Figure 4. Varied MI Values of Quartz

Figure 5 shows us the non-linear behavior and urges the appeal of standardization of MI tools. If the software system needs to be evaluated using these tools, how can we articulate that a particular tool can be used to calculate the maintainability of the software system in ideal approach?

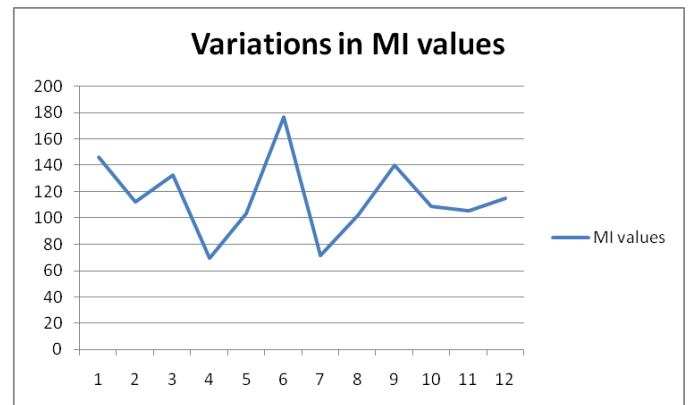


Figure 5. Unpredicted Variation in MI values

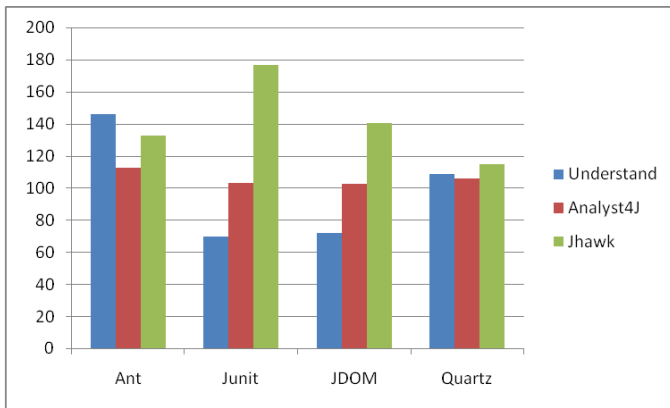


Figure 6. Varying Behavior of MI Tools for the Selected Software Systems

Figure 6 gives a comprehensive overview of the discussion being made, and exploits the fame of MI tools as none of them was able to synchronize its result with others and this creates an ambiguity to determine which of the software system should be called maintainable in the jargon of MI tools.

IV. CONCLUSION

As the MI gets more and more accepted in the software industry, it is desirous to have analysis of the available MI calculation tools, explore and test their functionality against different test scenarios, and suggest standardization. We have presented the analysis that the available MI tools provide somewhat idea of software maintainability but they all lack in providing acceptable, credible, and unified results as the outcome is situation dependent and is according to a fixed built in functionality, see section III. The tools vary in procedures of calculating different parameters, some tools focus on one parameter which is ignored by the other. There is a strong need to standardize the way MI is being calculated.

In order to standardize MI tools it will be a requirement that all the tools follow the best techniques in calculating different parameters of MI. As the results of various software system's MI values have shown in this paper, we are unable to recommend which of the available MI tools can be aspirant for prediction of correct MI values.

The study clearly shows that there is a strong need for making some standardization to calculate MI values. This standardization will help to calculate MI in a more pragmatic manner. This kind of MI calculation tool will go a long way helping the software industry in achieving better, maintainable, acceptable and quality software. This will help tools to be unified in estimating the maintainability of software systems.

REFERENCES

- [1] M. Imran Sarwar, Wasif Tanveer, Imran Sarwar, Dr. Waqar Mahmood, "A Comparative Study of MI Tools: Defining the Roadmap to MI Tools Standardization", IEEE International Multitopic Conference (IEEE INMIC 2008), Bahria University, Karachi, Pakistan December 23-24, 2008.
- [2] Coleman, D.; Lowther, B.; and Oman, P.; Using Metrics to Evaluate Software System Maintainability, IEEE Computer, Vol. 27(8), pp. 44-49, Aug. 1994.
- [3] C. M. Software Engineering Institute, "Maintainability Index Technique for Measuring Program Maintainability – software technology roadmap," http://www.sei.cmu.edu/activities/str/descriptions/mitmpm_body.html.
- [4] [IEEE 90] Institute of Electrical and Electronics Engineers. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York, NY: 1990. http://www.sei.cmu.edu/str/indexes/references/IEEE_90.html
- [5] Halstead, Maurice H. Elements of Software Science, Operating, and Programming Systems Series Volume 7. New York, NY: Elsevier, 1977.
- [6] McCabe, Thomas J., "A Complexity Measure," IEEE Transactions on Software Engineering, SE-2 No. 4, pp. 308-320, December 1976.
- [7] Szulewski, Paul, et al. Automating Software Design Metrics (RADCTR-84-27). Rome, NY: Rome Air Development Center, 1984.
- [8] Al-Khwarizmi Institute of Computer Science, University of Engineering and Technology, Lahore. - 'An Institution Dedicated to Algorithms' <http://www.kics.edu.pk>
- [9] Coleman, D., "Assessing Maintainability," 1992 Software Engineering Productivity Conference Proceedings, Hewlett-Packard, 1992, pp. 525-532.
- [10] blog dds: 2005.02.04 - "Maintainability of the FreeBSD System" <http://www.spinellis.gr/blog/20050204/index.html>
- [11] Oracle FAQ – "Engineering better PL / SQL" <http://www.orafaq.com/node/846>
- [12] Open Source Initiative <http://www.opensource.org/>