Proc. EECSI 2020 - 1-2 October 2020

Comparison of Maintainability Index Measurement from Microsoft CodeLens and Line of Code

Gilang Heru Kencana
Electrical Engineering Departement
Politeknik Elektronika Negeri Surabaya
Surabaya, Indonesia
gilangkencana@te.student.pens.ac.id

Akuwan Saleh
Electrical Engineering Departement
Politeknik Elektronika Negeri Surabaya
Surabaya, Indonesia
akuwan@pens.ac.id

Haryadi Amran Darwito
Electrical Engineering Departement
Politeknik Elektronika Negeri Surabaya
Surabaya, Indonesia
amran@pens.ac.id

R. Rizki Rachmadi
Electrical Engineering Departement
Politeknik Elektronika Negeri Surabaya
Surabaya, Indonesia
rizkirachmadi@te.student.pens.ac.id

Elsa Mayang Sari
Electrical Engineering Departement
Politeknik Elektronika Negeri Surabaya
Surabaya, Indonesia
elsamygs@te.student.pens.ac.id

***Abstract*—Higher software quality demands are in line with software quality assurance that can be implemented in every step of the software development process. Maintainability Index is a calculation used to review the level of maintenance of the software. MI has a close relationship with software quality parameters based on Halstead Volume (HV), Cyclomatic Complexity McCabe (CC), and Line of Code (LOC). MI calculations can be carried out automatically with the help of a framework that has been introduced in the industrial world, such as Microsoft Visual Studio 2015 in the form of Code Matric Analysis and an additional software named Microsoft CodeLens Code Health Indicator. Previous research explained the close relationships between LOC and HV, and LOC and CC. New equations can be acquired to calculate the MI with the LOC approach. The LOC Parameter is physically shaped in a software program so that the developer can understand it easily and quickly. The aim of this research is to automate the MI calculation process based on the component classification method of modules in a rule-based C # program file. These rules are based on the error of MI calculations that occur from the platform, and the estimation of MI with LOC classification rules generates an error rate of less than 20% (19.75 %) of the data, both of which have the same accuracy.**

Keywords— Software Quality, Maintainability Index, Halstead Volume, Cyclomatic Complexity, LOC.

I. INTRODUCTION

According to ISO / IEC 25010:2011 standardization, there are 8 main characteristics for the description of the product quality model: functional suitability, reliability, performance efficiency, usability, security, compatibility, maintainability and portability. In addition, the standard sets out the relationship metrics that can be used as indicators of these characteristics [1].

In particular, Oman et al. proposed the Maintainability Index (MI) [2][7], Numerous software metrics have been proposed as standards for software quality products. Maintainability is the level of efficiency and effectiveness where the product or system could be altered by the intended implementers [1]. MI is a composite matrix of multiple source metric codes in a single unified value of the maintainability indicator. As for Halstead's Volume (HV), McCabe's Cyclomatic Complexity (CC) and Line of Code (LOC), they are designed to measure the parameters of the MI. MI is the most widely used metric in the industry and has been successfully implemented in software systems such as Visual Studio [3]. In this research, MI calculations are used in Visual Studio using both Code Matric and the Visual Studio extension, which is Microsoft CodeLens.

This study obtained approximate expressions from MI metrics generated by visual studio and LOC-dependent measurements. It needs to detect anomaly from the results of the MI. The unexpected LOC value from the data code metrics, such as the identifier variable and the defined variable, can be evaluated by reviewing the MI. The exclusion process is used to have a new LOC value to improve the suitability of the MI value calculated from the closed calculations with the calculation derived from the LOC.

The paper is organized as follows. The connected works are explained in Section 2. The projected technique is granted in section 3. The experimental results of the expected method of analysis with others are presented in section 4. The conclusion is in section 5.

II. RELATED WORKS

A lot of research has been done on modeling maintenance software in measurement for the development and improvement of software systems. From the 1970s to 2020, a number of predictive models or measurement techniques have been developed. In [4], Berns concluded that the maintenance factor depends on the scale of complexity of recognizing the software program. Measurement of MI problems in Visual Studio is performed by evaluating the values of HV, CC and LOC. No rules are in place to re-evaluate the LOC value that was assessed from the initial calculation. The steps for calculating the flow are shown in Fig 1.

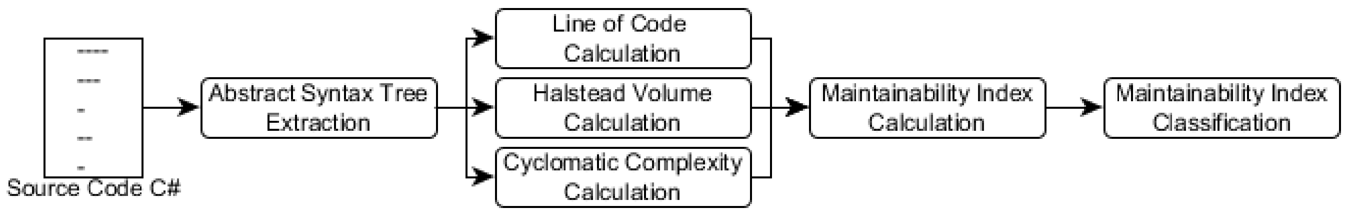
Source Code C#
Abstract Syntax Tree Extraction
Line of Code Calculation
Halstead Volume Calculation
Cyclomatic Complexity Calculation
Maintainability Index Calculation
Maintainability Index Classification

Fig. 1. Calculation and Classification of Maintainability Index

There are several steps to calculate the value of the Maintainability Index using the C # source code. The calculation stage starts with extracting the abstract syntax tree C # source code to find the operands, the operators, the number of lines of the source code and the number of lines of the comments. Operands and operators that have been identified will be used as input for the calculation of HV and CC. The results will be used as input to the MI formula after calculating the values of HV and CC. Once the MI value has been known, it would be classified on the basis of its value. There are three types of classifications, namely High Maintainability, Moderate Maintainability and Low Maintainability, as shown in Table 1.

MI is a value based on the HV metric[5], the CC metric[6], the average number of LOCs per module, and the average number of comments per module (COM). The HV metrics depend on the unit metrics that support the number of operators and operands in the source code. For the CC metric, the result measurement of the separation model is from the program body and the subroutines and functions, and then the CC of the source code is determined by summing up the CCs per module. The derivatives used by SEI should be calculated as follows[7]:

$$\mathrm{MI} = 171 - 5.2\times\ln(\mathrm{HV}) - 0.23\times\mathrm{CC} - 16.2 \times\ln(\mathrm{LOC})+50\times\sin\sqrt{(2.4\times\mathrm{COM})} \qquad (1)$$

The derivative used by Microsoft Visual Studio (since 2008) is calculated as follows [11]:

$$\mathrm{MI} = \mathrm{MAX}(0, (171 - 5.2\times\ln(\mathrm{HV}) - 0.23\times\mathrm{CC} - 16.2\times\ln(\mathrm{LOC}))*100/171) \qquad (2)$$

The higher the MI, the easier it is to maintain the software system. Measurements used in the open-source metric software measurement platform include the Microsoft Visual Studio 2019 development field and the Microsoft CodeLens Health Indicator, which ranges from 0 to 100[12].

TABLE I. CLASSIFICATIONS MAINTAINABILITY INDEX

Maintainability Index	**Classification**
$\mathrm{MI} \geq 20$	Green (High Maintainability)
$10 \leq \mathrm{MI} < 20$	Yellow (Moderately Maintainability)
$\mathrm{MI} < 10$	Red (Low Maintainability)

III. PROPOSED METHOD

The method is proposed by referring to the relationship of the calculation of the maintainability index between the metrics code and the LOC calculation. The approach between the two intended input parameters is to be used as a reference to the quality assessment of the software. The activity diagram of the proposed method is shown in Fig 2.

After entering the source code of C #, the program is inserted into the analytical code of the metrics. The metrics obtained in Visual Studio 2015 are as follows:

1. Maintainability Index
2. Cyclomatic Complexity
3. Depth of Inheritance
4. Class Coupling
5. Lines of Source Code
6. Lines of Executable Code

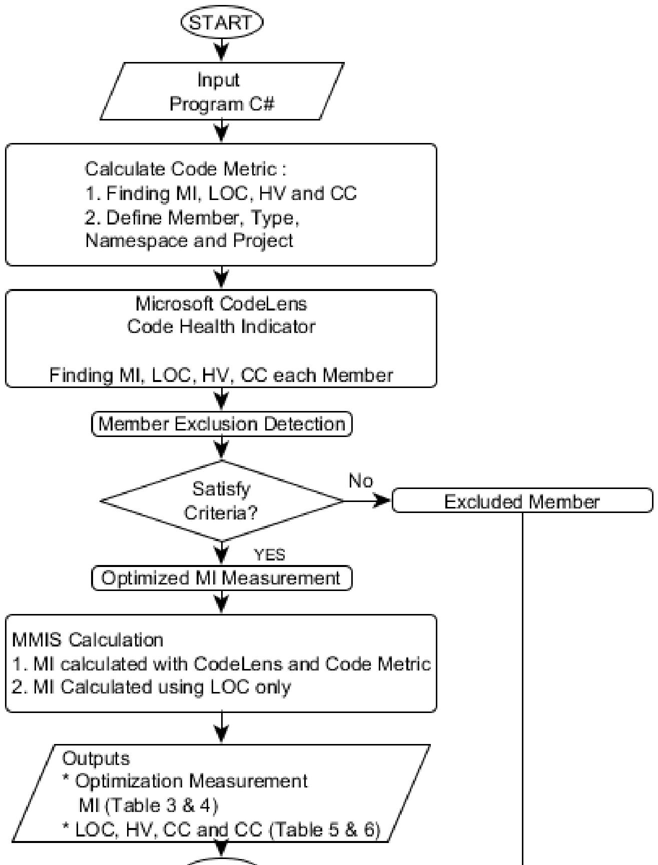
START
Input
Program C#
Calculate Code Metric :
1. Finding MI, LOC, HV and CC
2. Define Member, Type, Namespace and Project
Microsoft CodeLens
Code Health Indicator
Finding MI, LOC, HV, CC each Member
Member Exclusion Detection
Satisfy Criteria?
No
Excluded Member
YES
Optimized MI Measurement
MMIS Calculation
1. MI calculated with CodeLens and Code Metric
2. MI Calculated using LOC only
Outputs
* Optimization Measurement MI (Table 3 & 4)
* LOC, HV, CC and CC (Table 5 & 6)
END

Fig. 2. Activity Diagram of Proposed Method

According to Equation 2, only some type of data from the results of the analysis can be used as a calculation of the metric code. The calculation of the metric code performed by Visual Studio 2015 is done directly and precisely. It is therefore necessary to re-evaluate the results of the Halstead Volume not found in the metrics code calculated as follows:

$$\mathrm{HV} = e^{((\mathrm{MI}\times\frac{171}{100} - 171 + 0.23\times(\mathrm{CC}) + 16{,}2\times\ln\mathrm{LOC})\times 10/-52)} \qquad (3)$$

After all data on the MI calculation is obtained, the MI Classification Measurement for each module can be continued. In the proposed method, the Satisfied Criteria will be followed by a comparison of the MI value derived from the metric code and the MI value derived from the LOC.

3.1. Maintainability Index Equation

A description of the Maintainability Index calculation method is discussed in this section.

3.1.1. The MI Code Metric Equation

Several attempts have been made to quantify the maintenance of a software system [8]. MI is a software metric that measures the level of maintainability of the source code based on the equation (2).

3.1.2. The MI Equation depending on LOC only

The main thing in the MI formula is the Line of Code, where the value of the LOC is discussed endlessly. The aim of this research is to find the reason why LOC has become the most critical thing, such as:

a. Calculation of Automation

LOC is a physical component, the manual rest can be replaced by an automated calculation. Several platforms have been able to perform calculations in specific programming languages, since each programming language has its syntax and structural differences.

b. Shape of Matric

LOC can serve as a matrix that has a specific size because it can be viewed in a physical form so that it can be visualized to express the size of the system software in a programming language and to evaluate it in a logical space observation. Based on some approach of the MI value with the LOC, we only advise the equation of the MI calculation with the LOC value. We need to know that in order to gain new equations [9].

$$\mathrm{HV} = |(45 \times \mathrm{LOC} - 428)| \quad (4)$$
$$\mathrm{CC} = 0.22 \times \mathrm{LOC} + 1.9 \quad (5)$$

By substituting the value of HV in equation (4), CC in equation (5) in formula MI in equation (2), we conclude a new MI equation.

$$\mathrm{MI} = \mathrm{MAX}(0, (171 - 5.2\times\ln(\mathrm{HV(eq.4)}) - 0.23\times (\mathrm{CC(eq.5)}) - 16.2 \times \ln(\mathrm{LOC})) \times 100/171) \quad (6)$$

3.2. Detection of Member Exclusion

Detection of Member Exclusion is used to process MI sorting, which is the error data (defects) of each member to increase the MI calculation error. Case studies are conducted by reviewing three variables of the MI members with the following conditions:

- $\mathrm{MI} \geq 0$ or $\mathrm{MI} < 100$
- $\mathrm{HV} \geq 1$ or $\mathrm{LOC} \geq 1$
- Not an identifier variable or re-defined variable

If the member fails to comply with any of the three requirements, the exception will be executed on the metrics code for that module. This exception occurs when the module is an identifier or a re-definition module. It is not the core or source of the program, so the value of LOC and Halstead Volume tends to be high because the number of LOC identifiers is defined by the member.

IV. Experimental Results

The experiments are conducted using the Intel Core i7 2.5 GHz CPU, 8 GB RAM, and Microsoft Windows 10 Home Single Language 64-bit (10.0, Build 18362) computing platform. The field of development in Visual Studio 2015, C # programming language, and E-Commerce Application Development. The dataset used in this study is from the Code Metric Visual Studio 2015, Microsoft CodeLens Code Health Indicator VS 2015 Extension Pack.

TABLE II. Code Metric Software and the Code Attributes

Visual Studio Code Metric				
Scope	**Type**	**MI**	**CC**	**LOC**
Project	(Sum All Type)	67	137	1445
Type	DashboardForm	44	76	1089
Type	Enroll	59	15	106
Type	Global	91	11	15
Type	Info	66	8	46
Type	Program	81	1	3
Type	SettingForm	57	21	165
Type	Success	68	5	21

In this experiment, the types of project dataset (DashboardForm, Enroll, Global, Info Program, SettingForm, and Success) are used. Each type of value is a representation of all kinds of members. As shown in Table 1, the project section of the Maintainability Index is derived from the average (CC, LOC, and HV), but there is no HV value in the initial dataset of the Code Metric. The amount of HV generated in Equation 3.

First, the MI Method of Measurement is carried out by Code Matric and CodeLens. The experimental results are shown in Table 3, Table 4 and Fig 3. The result is a relatively small error rate of 10%, with the exception of some members pursuant to section 3.2. Optimized MI Measurement results show that the exceptions are closer to each other than those in Table 2 datasets.

TABLE III. Optimized MI Measurement (Visual Studio Code Metric)

Type	**Visual Studio Code Metric**			
	LOC	**HV**	**CC**	**MI**
(Sum All Type)	581	12256	125	61.8
DashboardForm	240	4338	75	64.4
Enroll	106	2260	15	59.6
Info	46	937	8	65.2
Program	3	16	1	81.0
SettingForm	165	4524	21	56.9
Success	21	181	5	68.9

TABLE IV. Optimized MI Measurement (CodeLens)

Type	**Microsoft CodeLens**				**% Error MI**
	LOC	**HV**	**CC**	**MI**	
(Sum All Type)	566	31480	125	61.2	0.885
DashboardForm	240	8965	75	63.6	1.252
Enroll	99	6560	15	57.0	4.350
Info	44	3204	8	61.8	5.090
Program	3	69	1	76.6	5.454
SettingForm	161	11772	21	54.2	4.703
Success	19	910	5	64.9	5.758

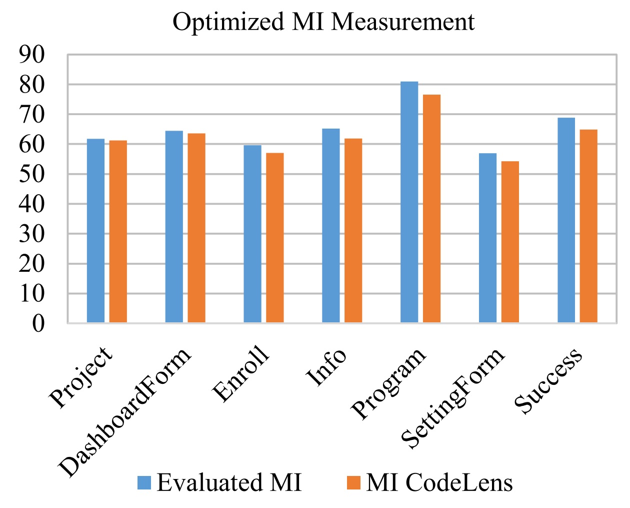

Fig. 3. Optimized MI Measurement on Code Metric and CodeLens

In the next experiment, the measurement of MI depends on LOC only. The experimental results are shown in Table 5, Table 6 and Fig 4. The proposed calculation of the method is comprehensible, fast to calculate, and independent of the programming language [10]. Results show that there was no unfortunate result when the LOC-based MI measurement was used.

TABLE V. COMPARISON MEASUREMENT MI FROM CODELENS AND LOC APPROACH (CODELENS)

Type	Microsoft CodeLens			
	LOC	HV	CC	MI
(Sum All Type)	566	31480	125	61.2
DashboardForm	240	8965	75	63.6
Enroll	99	6560	15	57.0
Info	44	3204	8	61.8
Program	3	69	1	76.6
SettingForm	161	11772	21	54.2
Success	19	910	5	64.9

TABLE VI. COMPARISON MEASUREMENT MI FROM CODELENS AND LOC APPROACH (LOC)

Type	MI on LOC Approach				% Error MI
	LOC	HV	CC	MI	
(Sum All Type)	566	24629	231	61.8	0.88
DashboardForm	240	8669	101	63.6	0.026
Enroll	99	3805	38.9	58.3	2.228
Info	44	3062	21.1	61.7	0.252
Program	3	293	2.56	72	6.401
SettingForm	161	7877	58.2	55	1.462
Success	19	923	9.88	64.7	0.405

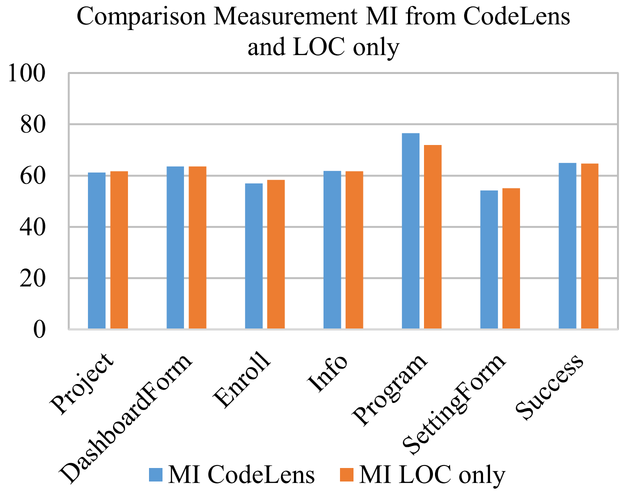

Fig. 4. Comparison MI Measurement

Indeed, it is necessary to know the number of differences when using the MI calculation based only on LOC and using the extended version of Microsoft Codelens when compared with the MI of the estimate. The data in Fig 5 is the result of a comparison between LOC and MI by CodeLens. In the meantime, if a manual approach is used, the plot will be shown in Fig 6.

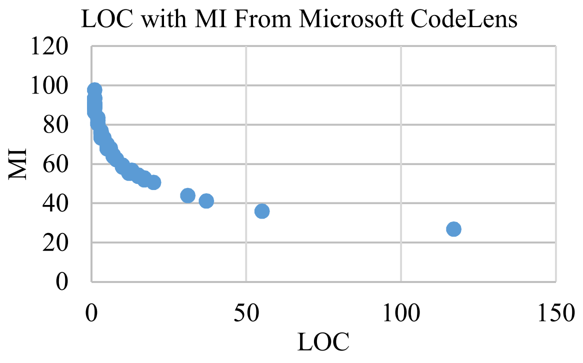

Fig. 5. Density Plot of LOC/MI from Microsoft CodeLens

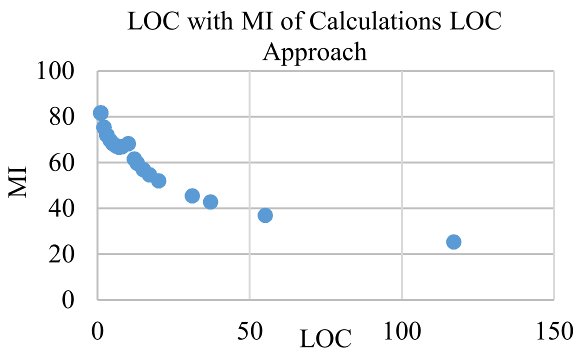

Fig. 6. Density Plot Of LOC/MI from LOC Approach

The plotting of the difference between these two methods will generate a maximum error lower than 20% (19.75%), which means the system between these two is already accurate.

V. CONCLUSION

The calculation of MI based only on the LOC approach is proposed to get easy to understand, quicker count for Measurement Maintainability Index for Software Quality. Implementation of Method Measurement Maintainability Index System could handle the optimization of values differences of HV, CC, and LOC. This technique is used to overcome the imbalance of the results of the LOC itself, while LOC could serve as a reference Software Quality, even Software Cost Effort Estimation. The purpose method is applied to the E-commerce project code metric data set with the context of Software Quality. The experimental results indicate that the technique achieved higher similarity accuracy on the other measurement of MI. In conclusion, the proposed method makes an improvement in MI calculation performance.

REFERENCES

[1] ISO/IEC, “ISO/IEC 25010:2011 - Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models,” 2011.

[2] P. Oman and J. Hagemeister, “Metrics for assessing a software system’s maintainability,” Proc. - Conf. Softw. Maintenance, ICSM 1992, pp. 337–344, 1992, doi: 10.1109/ICSM.1992.242525.

[3] C. Chen, R. Alfayez, K. Srisopha, B. Boehm, and L. Shi, “Why is it important to measure maintainability and what are the best ways to do it?,” Proc. - 2017 IEEE/ACM 39th Int. Conf. Softw. Eng. Companion, ICSE-C 2017, pp. 377–378, 2017, doi: 10.1109/ICSE-C.2017.75.

[4] G. M. Berns, “Assessing software maintainability,” Commun. ACM, vol. 27, no. 1, pp. 14–23, 1984, doi: 10.1145/69605.357965.

[5] M. H. Halstead, Elements of Software Science (Operating and Programming Systems Series). USA: Elsevier Science Inc., 1977.

[6] T. J. McCabe, “A Complexity Measure,” IEEE Trans. Softw. Eng., vol. SE-2, no. 4, pp. 308–320, Dec. 1976, doi: 10.1109/TSE.1976.233837.

[7] D. Coleman, D. Ash, B. Lowther, and P. Oman, “Using Metrics to Evaluate Software System Maintainability,” Computer (Long. Beach. Calif)., vol. 27, no. 8, pp. 44–49, 1994, doi: 10.1109/2.303623.

[8] A. Ganpati, A. Kalia, and H. Singh, “A Comparative Study of Maintainability Index of Open Source Software,” Int. J. Emerg. Technol. Adv. Eng., vol. 2, no. 10, pp. 228–230, 2012, [Online]. Available: http://www.ijetae.com/files/Volume2Issue10/IJETAE_1012_40.pdf.

[9] M. J. P. Van Der Meulen and M. A. Revilla, “Correlations between internal software metrics and software dependability in a large population of small C/C++ programs,” Proc. - Int. Symp. Softw. Reliab. Eng. ISSRE, no. May 2004, pp. 203–208, 2007, doi: 10.1109/ISSRE.2007.12.

[10] N. M.A.M.Najm, “Measuring Maintainability Index of a Software Depending on Line of Code Only,” IOSR J. Comput. Eng., vol. 16, no. 2, pp. 64–69, 2014, doi: 10.9790/0661-16276469.

[11] projectcodemeter, “Maintainability Index (MI),” Maintainability Index (MI) | projectcodemeter. [Online]. Available: http://www.projectcodemeter.com/cost_estimation/help/GL_maintainability.htm. [Accessed: 01-Jun-2020].

[12] Mike Jones, “Calculate code metrics - Visual Studio,” Calculate code metrics - Visual Studio | Microsoft Docs. [Online]. Available: https://docs.microsoft.com/en-us/visualstudio/code-quality/code-metrics-values?view=vs-2019. [Accessed: 01-Jun-2020].