LETTER

Empirical Evaluation of Method Complexity for C++ Program

Motoyasu TAKEHARA†, Toshihiro KAMIYA†, *Nonmembers*, Shinji KUSUMOTO†, *and* Katsuro INOUE†,††, *Regular Members*

SUMMARY This letter empirically evaluates the way how to calculate the complexity of methods, that is used in the definition of WMC(Weighted Method per Class), one of the Chidamber and Kemerer's metrics. With respect to the results of our experiment, Halstead's Software Science metric is the most appropriate one to evaluate the complexity of the methods.

key words: *metrics, object-oriented software, complexity, C++*

1. Introduction

WMC(Weighted Method per Class) is one of the Chidamber and Kemerer's metrics (C&K metrics) which are the most well-known complexity metrics for object-oriented software. WMC measures the complexity of the target class and is defined as the total complexity of the methods included in the class. However, what is the complexity of a method has not been defined [3].

This letter empirically evaluates the way to calculate methods' complexity in the class using the data collected from an actual object-oriented software development process.

2. WMC of C&K Metrics[3]

The definition of WMC is as follows:

Consider a class C, with methods $M_1,\ldots, M_n$ that are defined in the class C. Let $c_i,\ldots, c_n$ be the complexity of the methods. Then,

$$WMC = \sum_{i=1}^{n} c_i.$$

Here, if all method complexities are considered to be unity, then WMC $= n$, the number of methods.

Several research studies have empirically evaluated the usefulness of C&K metrics [1], [2]. For example, Basili et al. empirically evaluated that C&K metrics show to be better predictors than the best set of traditional code metric. In the evaluations, WMC was simply measured with the number of methods in each class.

Manuscript received September 22, 1999.

†The authors are with Graduate School of Engineering Science, Osaka University, Toyonaka-shi, 560–8531 Japan.

††The author is with Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma-shi, 630–0101 Japan.

Generally, since the size, algorithm and data structure of a method affect the complexity of it, it is suitable to calculate the WMC based on the characteristics of the size, algorithm and data structure.

3. Candidates for WMC

Here, we prepare the following eight candidates of WMC, each of them uses traditional metric for the size, algorithm and data structure as the complexity of a method, respectively. Each of the definition is as follows:

- WMC1: c_i is defined as the number of lines of M_i.
- WMC2: c_i is defined as the number of lines of M_i, except comments and empty lines.
- WMC3: c_i is defined as the number of unique variables referred by M_i.
- WMC4: c_i is defined as the total number of variables referred by M_i.
- WMC5: c_i is defined as the number of unique functions called by M_i.
- WMC6: c_i is defined as the total number of functions called by M_i.
- WMC7: c_i is defined as the McCabe's cyclomatic number of M_i.
- WMC8: c_i is defined as the Halstead's metrics of M_i.

For WMC7, cyclomatic number was introduced by McCabe to quantify control flow complexity [6]. It derived from the graphic representation of a program's control flow. The node in the graph representation corresponds to a decision or target in the program. Cyclomatic number equals to the number of disjoint regions.

For WMC8, it is based on the Halstead's observation that any computer program can be viewed as a sequence of tokens that can be classified as either operators or operands [4]. The basic metrics of it are, n_1: number of unique operators, n_2: number of unique operands, N_1: total number of operators, and N_2: total number of operands. Based on the basic metrics, Halstead defined several metrics for length, volume, level of abstraction and effort. Here, we use the following one of the Halstead's metrics for effort estimation:

$$c_i = \frac{1}{18}\frac{((N_1+N_2)\log_2(n_1+n_2))^2}{(2+n_2^*)\log_2(2+n_2^*)}.$$

Table 1 Type of components.

Component	Content	Num.
String	Operation of string list	10
List, ListNode	Operation of queue of string list	8
BigInteger	Operation of high number	1
TextAligner	Orrangementof the position of word	1
Sorter	Sorting data	13

Also, for convenience, we use WMC0 that is the number of methods in the class.

4. Empirical Evaluation

In order to evaluate which WMC is the most appropriate one, we apply these WMCs to an experimental software development project.

4.1 Outline of Experiment

The experimental project was performed in a computer company. The main characteristics of the project are: (1) Subjects are new employees of the computer company, who have just graduated from college. (2) Each subject selects one class (component) out of six classes shown in Table 1 and codes it using Visual C++. (3) Specification for each class is described by an instructor. Especially, the interface of each class is clearly defined. Thus, the complexity of each class depends on the implementation of the methods in it. (4) Each class is finally tested by the instructor.

4.2 Empirical Data

In the experiment, we finally collected data from 41 classes. The experimental data is shown in Table 2. We calculated the value of each WMC based on the final program code. As space is limited, Table 2 shows the maximum, minimum, average and standard deviation for each WMC.

Table 2 also shows the value of modifications of the classes. It represents the total number of lines which were influenced by changing the program during the program development. It is reported that the quantity of modifications correlates closely with the number of faults that were injected and removed from the program [5]. Thus, we consider that the quantity of modifications indirectly represents the complexity of the class and it is appropriate to evaluate the WMCs.

Table 2 Empirical data.

	Max.	Min.	Average	std. dev.
WMC0	34	7	19.26	9.09
WMC1	897	23	305.36	186.36
WMC2	751	23	278.24	165.30
WMC3	23	11	66.80	31.93
WMC4	612	11	281.51	172.84
WMC5	139	0	62.31	46.27
WMC6	190	0	86.21	66.30
WMC7	174	7	64.68	40.83
WMC8	28892	9	7431.94	6111.53
Modifications	84	0	15.90	18.31

4.3 Analysis

Table 3 shows the correlation coefficient between each of WMCs and the modifications. Most candidates for WMC are higher correlation with the modifications than WMC0. As the results, these WMCs can evaluate the complexity of the class better than WMC0. Especially, WMC8 which uses Halstead's metric as complexity of the methods, is highest correlation with the modifications. As the results, in this experiment, Halstead's metric is the most appropriate to evaluate the methods' complexity of WMC.

Table 3 Correlation coefficient.

WMC0	WMC1	WMC2	WMC3	WMC4	WMC5	WMC6	WMC7	WMC8
0.44	0.65	0.61	0.46	0.47	0.37	0.46	0.58	0.70

5. Conclusion

This letter has empirically shown that in calculating the WMC, it is appropriate to evaluate the complexity of the methods in the class instead of only counting the number of methods. Especially, WMC using Halstead's metric represents the internal complexity of the class more correctly.

As future research work, it is necessary to conduct the similar experiment for the data collected from an practical object-oriented software development process.

Acknowledgement

We would like to thank the support of Mr. Yukio Mohri and Mr. Yuichi Obata of Nihon Unisys corporation for the experimental project described in Sect. 4. Also, this reserach was supported in part by a grant from the Telecommunications Advancement Foundation.

References

[1] V.R. Basili, L.C. Briand, and W.L. Melo,"A validation of object-oriented design metrics as quality indicators," IEEE Trans. Software Eng., vol.22, no.10, pp.751–761, 1996.
[2] L.C. Briand, J. Daly, V. Porter, and J. Wust,"Predicting fault-prone classes with design measures in object-oriented systems," Proc. 9th ISSRE, pp.334–343, 1998.
[3] S.R. Chidamber and C.F. Kemerer,"A metrics suite for object oriented design," IEEE Trans. Software Eng., vol.20, no.6, pp.476–493, 994.
[4] V.Y. Shen and S.M. Thebaut, Halstead's Software Science, in Encyclopedia of Software Engineering, ed., J.J. Marciniak, vol.1, pp.534–535, John Wiley & Sons, 1994.
[5] S. Kusumoto, K. Matsumoto, T. Kikuno, and K. Torii, "On

a measurement environment for controlling software development activities," IEICE Trans., vol.E74, no.5, pp.1051–1054, May 1991.

[6] T.J. McCabe and C.W. Butler, "Design complexity measurement and testing," Commun. ACM, vol.32, no.12, pp.1415–1425, 1989.