Title | Empirical Evaluation of Method Complexity for C++ Program
Author(s) | Takehara, Motoyasu; Kamiya, Toshihiro; Kusumoto, Shinji; Inoue, Katsuro
Citation | IEICE Transactions on Information and Systems. E83-D(8) P.1698-P.1700
Issue Date | 2000-08-25
Text Version | publisher
URL | http://hdl.handle.net/11094/51052
DOI |
rights | copyright©2000 Institute of Electronics, Information and Communication Engineers

Osaka University Knowledge Archive : OUKA
https://ir.library.osaka-u.ac.jp/repo/ouka/all/

Osaka University
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 calculate 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 oriented software. WMC measures the 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 M₁, . . . , Mᵦ that are defined in the class C. Let c₁, . . . , cᵦ 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.

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

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, n₁: number of unique operators, n₂: number of unique operands, N₁: total number of operators, and N₂: 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:

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

#### 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. 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.

### 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
