IJCTA, 9(27), 2016, pp. 211-218 ISSN: 0974-5572
© International Science Press

An Analysis of Novel Temporal Coupling Factor Metric for Evaluating the Quality of Software

M. Kavitha* and S.A. Sahaya Arul Mary**

ABSTRACT

Temporal coupling in software programming refers to the dependency on series of actions between the modules over a particularinstance of time. The existence of high temporal coupling in software creates problems in terms of increased complexity and poor maintainability as the dependency between modules let a change in one module also let the changes in other connected modules. So far, there is no specific metric proposed to measure the level of temporal coupling involved in a module. Hence, the objective of this paper is to propose a novel Temporal Coupling Factor (TCF)metricto evaluate the level of the existence of temporal couplingin given software. The software with low temporal coupling is highly acceptable than the software with medium and high temporal coupling as the low temporal coupling have only less connected modules. The proposed metric has been empirically evaluated with a case study to calibrate the findings of the proposed TCF Metric.

Keywords: coupling, metric, maintainability, complexity, temporal coupling.

1. INTRODUCTION

Software metrics have been the part and parcel of the software evaluation process for assessing the quality of any software before its delivery. Software may assessed by variety of quality attributes such as complexity, maintainability and reusability. Among them, complexity and maintainability plays a vital role for assessing the worthiness of software as itincreases the scope of future enhancements. The more the complexity of the software increases is the more the scope of software maintainability decreases. There are plenty of software metrics presented in the literature for assessing the complexity and maintainability of software, with which the important one is the coupling metric.

Coupling refers to the degree of dependency between software modules and also measures the strength of relationships of modules within one another [1]. Software with high cohesion and low coupling is often desirable as the low coupling indicates that the built software consists of well-structured programs. Software with high coupling indicates the incurrence of high complexity in the software which would in turn affects software maintainability. The types of coupling can be majorly categorized into procedural and object oriented programming. There aretwo typesof object-oriented couplings possible as

- *Subclass coupling*: designates the dependency between the base and derived classes.
- *Temporal coupling*: when two or more actions are assimilated into one module just because they occur in a particular instance of time. For instance, processes whose activities are orderly sequenced or whose outputs are carry forward and waiting for a response to a prior request.

This paper analyzes one of the couplings of object oriented programming called temporal coupling. It is evident that the high temporal coupling degrades the performance of software in all aspects; hence, it is inevitable for the developers to identify the level of temporal coupling involved in the software. If the level of temporal coupling is

* Research Scholar, Dept. of Computer Science, Manonmaniam Sundaranar University, Tirunelveli.

** Dean Academics,Jayaram College of Engineering & Technology, Trichy.

high, then the developer must revise the program to fix it into a low temporal coupling by evading the unnecessary connections within modules.

The remaining section of the paper is organized as follows: section 2 describes the review of literature, section 3 designates, section

2. REVIEW OF LITERATURE

Újházi et al.[2] have presented a conceptual coupling between object classes (CCBO) metric from CBO coupling metric and empirically evaluated the CCBO metric for predicting the fault-proneness of classes in large open source system. The authors have analyzed and converted the source code into a corpus of textualdocuments where each document corresponds to the implementation of a method. A technique called LSI (Latent Semantic Indexing) takes the corpus as an input and creates aterm-by-document matrix, which captures the dispersionand co-occurrence of terms in class methods. SVD isused next to construct a subspace, referred to as the LSIsubspace. All methods from this matrix are representedas vectors in the LSI subspace. The cosine similaritybetween two vectors is used as a measure of conceptualsimilarity between two methods and is supposed todetermine shared conceptual information between twomethods in the context of the entire software system.The authors have also compared their metrics with the existing structural and conceptual metrics for the same fault-proneness task.

Yadav et al.[3] have applied Cohesion and Coupling metrics on programs of inheritance and interface and have evaluated the existing software metrics values. The cohesion and Coupling metrics identifies the complexity between inheritance and interface programming. The authors have showed that the proposed conceptis good to use. The authors have also focused on the empirical evaluation of object oriented metrics in C#. The resulting values are analyzed to provide significant insight about the object oriented characteristics of reusability programs.

Aloysius et al. [4] have presented a cognitive complexity metric namely cognitive weighted coupling between objects for measuring the types of coupling involved in object- oriented systems. The authors concentrated on five types of coupling that may exist between classes such as control coupling, global data coupling, internal data coupling, data coupling and lexical content coupling were considered in computing their proposed CWCBO. CWCBO had proven that, complexity of the class was getting affected based on the cognitive weights of the various types of coupling.

Poshyvanyk et al.[5] have introduced set of coupling measures for OO systems – named conceptual coupling, based on the semantic information obtained from the source code, encoded in identifiers and comments. The authors have conducted a case study on open source software systems to compare the new measures with existing structural coupling measures. They authors have also proved that their case study is able to capture the conceptual coupling in new dimensions of coupling, which are not captured by existing coupling measures.

Misra et al.[6] have proposed a metric for the inheritance properties of object oriented programming. The metric calculates the complexity at method level considering internal structure of methods, and also considers inheritance to calculate the complexity of class hierarchies. The proposed metric is validated both theoretically and empirically. For theoretical validation, principles of measurement theory are applied since the measurement theory has been proposed and extensively used in the literature as a means to evaluate the software engineering metrics. The authors have applied their metric on a real project for empirical validation and compared it with Chidamber and Kemerer (CK) metrics suite [7]. The theoretical, practical and empirical validations and the comparative study have proved the robustness of the measure.

Aggarwal et al.[8] have addressed the need for software metrics and introduced a new set of design metrics for object-oriented code. The authors have developed two OO metrics for measuring the amount of robustness included in the code. The metrics are analytically evaluated against Weyuker’s [12]axioms. These set of metrics are calculated and analyzed for standard projects and accordingly ways in which project managers could utilize these metrics are suggested by the authors.

3. COUPLING FACTOR (CF)

MOOD proposed a metric called Coupling Factor (CF) for assessing the ratio of coupling between the classes [9]. The ratio of coupling is estimated using a fraction, where the numerator represents the total number of non-inheritance couplings in the module and the denominator signifies the maximum number of coupling that is possible for the corresponding module. The maximum number of couplings includes both inheritance and non-inheritance related coupling.

$$CF = \frac{\sum_{i=1}^{n}\left[\sum_{j=1}^{m} is\_client\left(C_i, C_j\right]\right.}{n^2 - n}$$

where'n' is the total number of classes in the module. The value of is_client (C_i, C_j) is 1 iff the class $C_j$calls a method or attribute of C_i or otherwise 0.

4. MOTIVATION

The experiential study of several researches recommendsdeveloping modules with high coupling is more difficult to understand, to locate the origin of errors and to perform addition and modification of programs in the existing modules. Moreover, excessive temporal coupling between objects is disadvantageous to modular design as more testing is required to achieve reliable results. Hence, to conclude, a module with low coupling is desirable. Though CF metric only exhibits the occurrence of inheritance oriented coupling between the classes, the earlier findings of the author discusses the direct and indirect coupling between the classes of the module by proposing an metric called SCF [10]. Temporal coupling also plays a vital role in assessing the complexity of any module. As the trend of software programming is majorly rely upon object oriented there should some proposals wanted for the types of OO couplings.

5. TEMPORAL COUPLING FACTOR

The calibration of temporal coupling is measured by analyzing the sequences of methods that processes an individual variable throughout all the classes in a module. The TCF value is obtained by performing the fraction of cumulative usage of all variables by the methods of the module with the total number of possibilities of temporal coupling within the module. An adjacency matrix between all coupling classes is created to trace the series of actions that are invoked when the variables are initialized. Let Mod be a module with several of interconnected classes with well-defined methods a coupling classes adjacency matrix $CAM_{v \times m}$ can be derived using the notion CAM [i,j].

$$CAM[i,j] = \begin{cases} 1 \text{ if variable } i \text{ is used in method } j \\ 0 \text{ if variable } i \text{ is not used in method } j \end{cases}$$

From this matrix, the series of actions that is performed with all variables are identified to be processed further. The accumulation of all series of actions with regard to all variables is then performed to compute the fraction with the possible temporal coupling in the module. The TCF is calculated as:

Let 'a' be the sum of all variables that is processed by the methods of the module which is initialized with 0.

$$TC = \frac{\sum_{i=0}^{v} \sum_{j=0}^{m} a = a + 1\left(if\, AM\left[i, j\right] = 1\right.}{CCV \times CCM \; - \; VCC \times MPC}$$

Where 'v' is the total number of variables and 'n' is the total number of methods in the module. 'CCV' is the number of coupling class variables,'CCM' is the number of coupling class methods, 'VCC' is the number of variables in the child class and 'MPC' is the number of methods in the parent class.

$$TCF = \frac{\sum_{i=0}^{tcg} TC}{Total\ number\ of\ Coupling\ Groups}$$

Tcg is the total number of coupling groups between the classes.

The maximum and minimum value of TCF can be 0 and 1 respectively. The module that results the TCF value closer to 1 signifies a high temporal coupling, the TCF value that is closer to 0 signifies the low temporal coupling and the TCF value that centers around 0.5 denotes the medium level temporal coupling. Modules with high and medium temporal coupling may be focused for further depreciation to a low level for reducing the complexity in the module.

6. EXPERIMENTAL RESULTS OF TEMPORAL COUPLING FACTOR

In order to measure the level of Temporal Coupling in module, first of all the adjacency matrix of the design is developed to apply the proposed metric. A case study has been conducted to evaluate the step-by-step process of TCF metric. The case study contains three classes with two coupling groups.

Case Study: Module with three classes

C1 = {m1, m2, m3}

M1 = {b, c}

M2 = {a, b, c}

M3 = {a, b, c}

C2 inherits class C1 = {m4, m5, m6}

M4 = {d, e}

M5 = {a, c, d}

M6 = {a, c, d, e}

C3 inherits class C1 = {m7, m8}

M7 = {a, g, h}

M8 = {a, b, c, g, h}

The pictorial representation of the module with its coupling details are given in Fig. 1

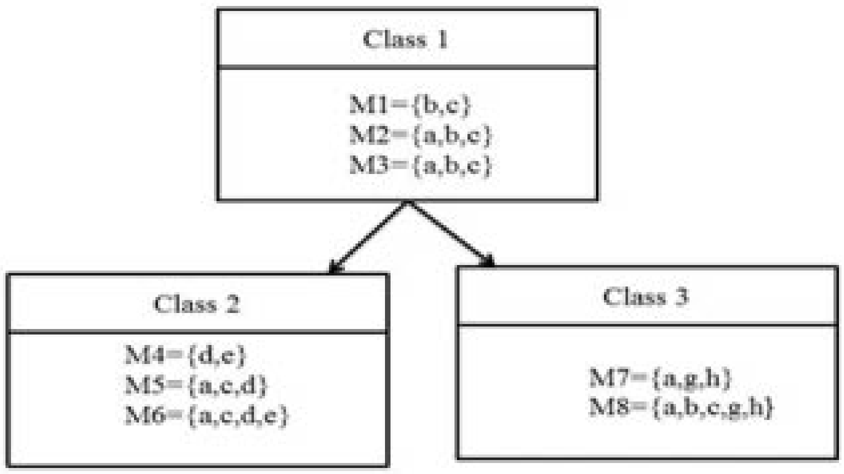

Figure 1: Class Diagram of Case Study

There are two coupling groups are seen the module such as i). Class1 with class 2 and ii). Class1 with class3, hence The CAM matrix for the first coupling group is computed as

$$CAM_1 = \begin{pmatrix} & m1 & m2 & m3 & m4 & m5 & m6 \\ a & 0 & 1 & 1 & 0 & 1 & 1 \\ b & 1 & 1 & 1 & 0 & 0 & 0 \\ c & 1 & 1 & 1 & 0 & 1 & 1 \\ d & 0 & 0 & 0 & 1 & 1 & 1 \\ e & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} sum \\ 4 \\ 3 \\ 5 \\ 3 \\ 2 \end{pmatrix}$$

The Temporal Coupling between the first coupled classes is the sum of usage of variables by the methods of coupled classes as

$$TC_1 = \frac{4+3+5+3+2}{(5\times 6)-(2\times 3)} = \frac{17}{24} = 0.71$$

The CAM adjacency matrix for the second coupling group is

$$CAM_2 = \begin{pmatrix} & m1 & m2 & m3 & m7 & m8 \\ a & 0 & 1 & 1 & 1 & 1 \\ b & 1 & 1 & 1 & 0 & 1 \\ c & 1 & 1 & 1 & 0 & 1 \\ g & 0 & 0 & 0 & 1 & 1 \\ h & 0 & 0 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} sum \\ 4 \\ 4 \\ 4 \\ 2 \\ 2 \end{pmatrix}$$

The Temporal Coupling between the second coupled classes is the sum of usage of variables by the methods of coupled classes as

$$TC_2 = \frac{4+4+4+2+2}{(5\times 5)-(2\times 3)} = \frac{16}{19} = 0.84$$

Total number of groups = 2

$$TCF = \frac{0.71+0.84}{2} = 0.78$$

The temporal coupling factor of the module is 0.78 which shows the existence of high temporal coupling level. The classes of the modules can be re-modified for reducing the complexity for future change.

7. TCF EVALUATION

Many inventions have suggested that the software metric should satisfy certain properties their real time usability in software testing. Basili and Reiter [13] suggested that software metrics should be sensitive to external observable differences in development process, and should correspond to intuitive notions about the characteristic differences between the software artifacts being measured. Weyuker has also proposed an authorized list of properties for software metrics that could be evaluated on the existing software metrics [14]. The notions of the Weyuker's properties include permutation, interaction, monotonicity, non- researchers have recommended various properties uniqueness and so on. The challenge in this section is to evaluate the proposed TCF coupling metric against the nine properties of Weyuker's to prove its usefulness.

Though, several Weyuker's properties are considered to be most significant to classify the complexity of a measure. Weyuker's properties state that

Property1

Non-coarseness

Not all class can have the same complexity. If there are 'n' numbers of modules in the software, TCF does not rank all 'n' modules as equally complex.

Property 2

Granularity: Let 'r' be a non-negative number and there could be only finite number of modules have the complexity r. If the number of modules in large scale system is finite, the complexity value of TCF is also finite. Hence this property is satisfied.

Property 3

Non-uniqueness: This property implies that there may be number of modules have the same complexity. TCF abides this property, if the hierarchies of class in the modules are similar.

Property 4

Design details are important

The property affirms that though if two classes have the same functionality, they may differ in terms of details of implementation. If the design implementation of two modules is different, TCF produces different complexity values for each module.

Property 5

Monotonicity

Let the concatenation of two modules R and S be R+S. Hence, this property states that complexity value of the combined class may be larger than the complexity of the individual classes R or S. TCF abides this property if there is a possibility of inheritance between the modules R and S while concatenation.

Property 6

Non-equivalence of interaction

This property states that if a new module is added to the two existing modules R and S which has the same module complexity, if a new module T is added with both modules, the module complexities of the two new combined modules may be different or the interaction between R and T may be different than the interaction between S and T resulting in different complexity values for R + T and S + T. TCF for sure yields different complexity values for both modules R and S since T is dependent on the fitness of inheritance with the existing modules R and S.

Property 7

Permutation: There are program bodies I and J such that J is formed by permuting the order of the statements of I and (|I| = |J|). This property is not taken into the consideration of object oriented metrics.

Property 8

Renaming

If module R is renamed as S then |R| = |S|. This property requires that renaming a module should not affect the complexity of the module. TCF does not have any impact over the change of name of module, hence TCF satisfies property 8.

Property 9

Interaction increases complexity

The property says that the class complexity measure of a new class combined from two classes may be greater than the sum of two individual class complexity measures. This property is not satisfied with TCF as the complexity of combined modules could be possibly equal to the individual complexity but not greater.

Summary of the TCF validation is described in Table 1.

Table 1
Evaluation of TCF Metric

Weyuker's Metric	P1	P2	P3	P4	P5	P6	P7	P8	P9
TCF	Y	Y	Y	Y	Y	Y	N	Y	Y

Where 'Y' represents Yes and 'N' represents No. As the proposed TCF metric satisfies eight out of nine properties of weyuker's, it is claimed as a valid metric to be used in software quality assessment.

8. CONCLUSION

Software metrics on temporal coupling on object-oriented concepts is one of the essential factorsto identify the quality of software in the testing process. This paper proposes a new software metric for assessing the level of temporal coupling involved in the software. The results of the TCF show that the higher level of temporal coupling increases the complexity. Moreover, thehigher complexity in software leads to more cost expensive and less maintainability of software. The assurance of less complexity software is of been great interest to researchers since the early days of development [11]. Hence, the proposed TCF metric will be helpful for the developers to identify the flaws in their program in the development stage itself.

REFERENCES

[1] https://www.infoq.com/news/2009/04/coupling

[2] Újházi, Béla, Rudolf Ferenc, Denys Poshyvanyk, and TiborGyimóthy. "New conceptual coupling and cohesion metrics for object-oriented systems." In*Source Code Analysis and Manipulation (SCAM), 2010 10th IEEE Working Conference IEEE,* 33-42. 2010.

[3] Yadav, Maya, Jasvinder Pal Singh, and PradeepBaniya. "Complexity Identification of Inheritance and Interface based on Cohesion and Coupling Metrics to Increase Reusability." *International Journal of Computer Applications* **64**, 31-35, 2013.

[4] Aloysius, A., and L. Arockiam. "Coupling complexity metric: A cognitive approach." *International Journal of Information Technology and Computer Science (IJITCS)* **4**, 29-35, 2012.

[5] Poshyvanyk, Denys, and Andrian Marcus. "The Conceptual Coupling Metrics for Object-Oriented Systems." In *ICSM*, 469-478, 2006.

[6] Misra, Sanjay, Ibrahim Akman, and Murat Koyuncu. "An inheritance complexity metric for object-oriented code: A cognitive approach." *Sadhana***36**, 317-337,2011.

[7] Chidamber, Shyam R., and Chris F. Kemerer. "A metrics suite for object oriented design." *IEEE Transactions on software engineering* **20**, 476-493, 1994.

[8] Aggarwal, K. K., Yogesh Singh, ArvinderKaur, and RuchikaMalhotra. “Software Design Metrics for Object-Oriented Software.” *Journal of Object Technology* **6**, 121-138,2007.

[9] Chou, Chen-huei. “Metrics in Evaluating Software Defects.” *International Journal of Computer Applications* **63**, 23-29, 2013.

[10] Mary, SA Sahaya Arul, and M. Kavitha. A Quality Based Novel Subclass Coupling Factor Metric for Evaluating Software.International Journal of Advanced Research in Computer and Communication Engineering, Vol. 5, Issue 3, March 2016. pp.643-647.

[11] Gill, Nasib S. “Dependency and interaction oriented complexity metrics of component-based systems.” *ACM SIGSOFT Software engineering notes* **33**, 1-5, 2008.

[12] Kumar, Rajnish. “Class complexity metric to predict understandability.” *International Journal of Information Engineering and Electronic Business***6**, 69-76,2014.

[13] Basili, Victor R., and Robert W. Reiter Jr. “Evaluating automatable measures of software development.” In *Proceedings on Workshop on Quantitative Software Models*, 107-116. 1979.

[14] Gandhi, Parul, and Pradeep Kumar Bhatia. “Analytical analysis of generic reusability: Weyuker’s Properties.” *International Journal of Computer Science* **9**, 424-427, 2012).