International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 5 | ISSN : 2456-3307

A Novel Temporal Cohesion Complexity Metric for Object-Oriented Design

Syed Tanzeel Rabani, S. Gayathri
Department of Computer Science, St. Joseph's College (Autonomous), Tiruchirappalli, Tamil Nadu, India

ABSTRACT

Software metric is used assess the quality of a software. There are many ways with which Software can be assessed but the predominant criterion would be the assessment of software code. Reusability is increased by the ease of maintenance of software code. Ease of maintenance also decreases the complexity in comprehending and understanding the program. Software program can be modularized based on various characteristics. The two measures Coupling and Cohesion can be used to access the quality of a design of these modules and their interaction. Cohesion refers to the degree of relationship between elements inside a module. Modules with high cohesion are preferred as they are associated with the reliability, maintainability, reusability and understandability of a software. No metric is available so far to determine the presence of temporal cohesion. Here, a novel attempt has been have made to evaluates the percentage of temporal cohesion involved in a module.

Keywords: Software metric, Coupling, Cohesion, Temporal cohesion (TC)

I. INTRODUCTION

Software engineering is an applied discipline of software science which acquires engineering approaches. It refers to the "application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software"[1]. The term "metrics" is used to denote a set of specific measurements taken on a particular item or process. Software Metri is defined as "the continuous application of measurement-based techniques to the software development process and its products to supply meaningful and timely management information, together with the use of those techniques to improve that process and its products" [2]. The primary objectives of the software metrics are to assess and to predict the quality of software. The main aim of Software metric is to verify the coding for the program. One of the most important and crucial segment of the software development is the coding where the quality can never be compromised. The programming can be done either through procedural oriented approach or by using object- oriented technology. Cohesion measures the strength of the functionality expressed by a software module, thus considered to be a justifying factor for measuring the quality of program code.

There is the possibility of six types of cohesion in a module viz coincidental, logical, temporal, procedural, communicational, sequential and functional cohesion. The quality of cohesion moves from coincidental to function cohesion. Temporal cohesion exists when parts of a module are grouped by when they are processed (i, e) they are processed at a specific time in program execution e.g. a function that is called after catching an exception which closes open files, creates a

Temporal cohesion is when parts of a module are grouped by as they are processed - the parts are processed at a particular time in program execution (e.g. a function which is called after catching an exception which closes open files, creates an error log, and notifies the user). As for now there is no specific metric available to identify whether a module is temporally cohesive or not. In this paper, a novel metric is proposed to measure percentage of temporal cohesion in a module to validate the quality of a product.

II. LITERATURE REVIEW

Gui et.al., [3] proposed two static metrics for cohesion namely WTCOh and WICoh to assess the reusability of java component. Authors proved that their metrics differ from majority of the cohesion metrics in three perspectives viz: they provide the measure to which entities are coupled or resemble each other, they quantify the indirect coupling and cohesion relationship and they also provide the information about functional complexity in classes and methods. By performing the empirical validation of the new metrics, the authors proved that that their metrics are better at measuring and ranking the reusability of software components.

Mann et al., [4] proposed two metrics TCC and LCC that were used to measure the design complexity of the software. The authors improved the applicability of the existing cohesion metrics to measure the requirement of refactoring of classes.

Kaur et al., [5] reviewed more than 20 class cohesion metrics in object oriented systems. This review was helpful to collectively gain knowledge and would lead to invention of several metrics such as Path Connectivity Class cohesion metric (PCCC), similarity-based Class Cohesion metric (SCC), Method- Method through attributes Cohesion Metrics (MMAC) in future. Desouky [6] proposed a metric called as RLCOM – DESOUKY metric that measures the degree of cohesion for objects of a class at runtime. These metrics were validated by correlating them to bugs, so that program behaviour could be observed during runtime.

Marcus et al., [7] proposed various measures such as C3, LSCM for the cohesion of Individual classes within an Object-Oriented System. The authors measured the proposed metrics using semantic information which was embedded in the source code. New measure was compared to the extensive set of existing measures by using the case study. Differences and similarities were discussed and analysed using existing and proposed metric. The proposed metrics help to identify the special cases like wrappers or classes that have the implementation of several concepts which can be refactored properly.

Hari Ganesh et al., [8] proposed the Coincidental-Functional Cohesion Metric that intended to assess whether the given module or class is coincidentally cohesive or functionally cohesive. CFCOM is calculated as follows:

$$\mathrm{CFCOM} = \frac{\sum \cap \cap_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{AM_i} \cap \mathrm{TAC}}{\mathrm{TAC}} \qquad (1)$$

Here, AM_i refers to the total number of attributes used in method i

TAC refers to the total number of Attributes defined in the class

The metric CFCOM functions by intersecting the variables of methods in a class with the variables that are defined in a class. The summation of the intersected variables is then divided by the overall possibilities that could be made within a class. Value 1 of CFCOM represents that the class is functionally cohesive, and Value 0 of CFCOM represents the class as coincidentally cohesive. Coincidentally cohesive class is an alarm for the programmers to redefine the class into an inseparable unit.

Hari Ganesh et al., [9] proposed the Sequential cohesion metric to evaluate the percentage of sequential cohesion involved in a module. SCOM is calculated as in Equation 2.

$$\mathrm{SCOM} = \frac{\sum_{\mathrm{i=}}^{\mathrm{n}} \mathrm{m_i} \cap m_i{+}1}{(\mathrm{n}-1\,)\times \mathrm{TAC}} \times 100\ \% \qquad (2)$$

Here, n' denotes the total number of methods in the module, 'mi' denotes ith method whereas mi ∩ mi+1 is the intersection of attributes of mi and mi+1, and TAC refers to the total number of attributes in a class. Strong sequential cohesion is determined by the 100% of SCOM whereas 0% denotes weak sequential cohesion. Hari Ganesh et al., [10] proposed the communicational cohesion metric (CCOM) for assessing the percentage wise communicational cohesion in software modules. The CCOM value of a module is given as under.

$$\mathrm{CCOM} = \frac{\mathrm{CM}}{\mathrm{CM}\ + \mathrm{NIVBM}} \times 100\% \qquad (3)$$

CM is the communicational measure which is derived by multiplying the sum of intersecting variables between methods by two and can be calculated below.

$$\mathrm{CM} = 2 \times \mathrm{IVBM} \qquad (4)$$

IVBM is calculated by the sum of Intersection of Variables between Methods which is calculated as below.

$$\mathrm{IVBM} = \forall_{i=1}^{n} \sum_{j=i+1}^{n} m_i \cap m_j \qquad (5)$$

Where, 'n' denotes the total number of methods in the module, 'm_i' and 'm_j' denotes ith and jthmethods whereas $m_i \cap m_j$ is the intersection of attributes of m_i and m_j. NIVBM represents the sum Non-intersecting of Variables Between Methods which is calculated as in Equation. 6

$$\mathrm{NIVBM} = \forall_{i=1}^{n} \sum_{j=i+1}^{n} !\,(m_i \cap m_j) \qquad (6)$$

A software module with the CCOM 100% value denotes a strong communicational cohesion and 0% value denotes weak communicational cohesion.

III. MOTIVATION

The poor design of program modules leads to the complexity of software and increase the cost of development. Maintenance is also costly for complex software. The use of metrics could reduce the feasible defects thus increasing the maintenance. Developing metrics to identify the highly cohesive code saves both time and cost for maintenance and reuse of project. Module acceptance also depends upon the type of cohesion, there is a need for inventing the new metrics to classify different types of cohesion in order to increase the quality of a software product.

IV. TEMPORAL COHESION METRIC

The proposed temporal cohesion metric is a novel metric that evaluates the percentage of temporal cohesion in a module. In Temporal cohesion statements are grouped together into a procedure and executed together during the same time-frame e.g. at the very beginning at the very end of a program. the code is put into a procedure and is executed because it is convenient to do so at a certain time in the program. Elements of a component in temporal cohesion is related by (4)timing. When any change (2.4)in data structure is made, it becomes difficult to look at numerous components thereby increasing the regression fault. Considering a module "On_Worst_Failure" that is invoked when a worst failure occurs. At that time, module performs several tasks that are not functionally similar or logically related, but all tasks need to happen at the moment of failure. The module (2.5) might

- Cancel all request for services
- Cut power to all machines
- Notify the operator about failure
- Make an entry in the database about the failure.

Temporal Cohesion (TC) is reserved for application specific, non-reusable code.

(2.6)

TC is defined as the percentage of the summation of coupling variables of the methods divided by the Total number of variables in methods contains expression.

Coupling variables (CV) = $\forall_{i=1}^{n}$ *Counter++ (if operand)*

Temporal cohesion (TC) = $\frac{\mathrm{CV}}{\mathrm{TVMCE}} \times 100$ (7)

Here, CV is the coupling variables and TVMCE is the total number of variables in methods contains expression.

A software possessing 100% of TC denotes a strong temporal cohesion and 0% denotes weak temporal cohesion. The implementation of temporal cohesion in software enhances the modularity of software program.

V. ILLUSTRATION

The Illustration of TC metric is evaluated against the three java programs which are described below.

# Example 1

```
import java.util.Scanner;
class VariousOperations
{
int a, b, diff, sum;
Scanner scan =new Scanner(System.in);
public void get()
{
System.out.println("Enter the values of A and B");
a=scan.nextInt();
b= scan.nextInt();
}
public void add()
{
sum = a+b;
}
public void diff()
{
diff = a-b;
}
public void disp()
{
System.out.println("Addition is" + sum);
System.out.println("Subtraction is" + diff);
}
}
class mainmethod
{
public static void main(String args[])
{
VariousOperations obj = new VariousOperations();
obj.get();
obj.add();
obj.diff();
obj.disp();
}
}
```

In example 1, class *VariousOperations* has two variables such as a and b and four methods namely get () add(), sum() and disp (). Therefore, the total number of attributes is 2 and the total number of methods (n) is 4. The method call is by calling the get () method is a input method and disp () method is a output method. The other two method add() and sum () is expression method. Total number of variables contains expression are two as follows:

Total number of coupling variables is (CV) {a,b} =2

m1= get () = {a, b} => Input Method
m2= add () = {a, b} => Expression Method
m3= sum () = {a, b} => Expression Method
m4= disp () = {a, b} => Output Method

Total Number of variables in methods contains expressions (TVMCE) = 2

Coupling variables

CV= $\forall_{i=1}^{n}$ *Counter ++ (if operand)*

CV =2;

Temporal cohesion (TC) = $\frac{CV}{TVMCE} \times 100$

TC= $\frac{2}{2} \times 100 = 100\%$

As the TC value of *various operations* program is 100%, the class is said to be full temporal cohesive.

# Example 2

```
import java.util.Scanner;
class Employee
{
int no;
float net;
String name;
Scanner scan =new Scanner(System.in);
public void get()
{
System.out.println("Enter the name");
name = scan.next();
System.out.println("Enter the number");
no = scan.nextInt();
}
public void sal()
{
System.out.print("Assign salary");
net = scan.nextFloat();
}
public void disp()
{
System.out.println("name is "+ name);
System.out.println("number is"+ no);
System.out.println("Salary is"+ net);
}
}
class mainmethodsec
{
public static void main(String args[])
{
Employee obj = new Employee();
obj.get();
obj.sal();
obj.disp();
}
}
```

In Example 2, Total number of coupling variables is (CV) {name, number, sal } =3

m1=get () = {no, name} => Input Method
m2=sal() = {sal} => Expression Method
m4=disp() = {no,name,sal}=> Output Method

Total Number of variables in methods contains expressions (TVMCE) = 1

Coupling variables

CV= $\forall_{i=1}^{n}$ *Counter ++ (if operand)*

CV =1;

Temporal cohesion (TC) = $\frac{CV}{TVMCE} \times 100$

$TC = \frac{1}{1} \times 100 \quad = 100\%$

# Example 3

```
import java.util.Scanner;
class square
{
double a;
Scanner scan =new Scanner(System.in);
public void sq1()
{
System.out.println("Enter the value for A");
a = scan.nextInt();
System.out.print("Square of a is" + a * a);
}
public void sq2()
{
System.out.println("enter the value for a again");
a = scan.nextInt();
System.out.println("square of a is " + a * a);
}
}
class mainmethod3
{
public static void main(String args[])
{
square obj = new square();
obj. sq1();
obj.sq2();
}
}
```

In Example 3, Total number of variables is {a} =1

m1=sq1 () = {a} => Expression Method

m2=sq2 () = {a} => Expression Method

Total Number of variables in methods contains expressions (TVMCE) = 2

Coupling variables

CV= $\forall_{i=1}^{n}$ *Counter ++ (if operand)*

CV =1;

Temporal cohesion (TC) = $\frac{CV}{TVMCE} \times 100$

$TC = \frac{1}{2} \times 100 \quad = 50\%$

The evaluated programs discussed above are compared with the results of standard LCOM metrics. The results are verified to check the enhancements of proposed metric with the LCOM

Table 1 : Comparison of Standard LCOM with TC

Program name	LCOM	TC
VariousOperations	0	100%
Employee	1	100%
Square	0	50%

The values 1 in LCOM represent only the existence of cohesion in methods, whereas the results TC more specifically represents the amount of Temporal cohesion involved in the program with an in-depth analysis of the program. Moreover, the results of LCOM do not precisely describe the differentiation on 1, but TC explicates that Employee is 100%, Square is 50% and VarriousOperation is 100% Temporal which would be useful for further acceptance or modification.

VI. ANALYTICAL EVALUATION OF TC

Property 1: Non-coarseness – (∃P) (∃Q) (|P| ≠ |Q|)

Not all class can have the same complexity. If there are ‘n’ numbers of classes in the module, TC does not rank all ‘n’ classes as equally complex. TC value of two different software’s complexity such as example 1 and example 2 are different from each other. Hence, this metric is satisfied.

Property 2: Granularity

Consider ‘a’ as a non-negative number then there co.uld be only finite number of classes and programs with complexity a. The value changes from one another; hence, only finite number of classes have the same complexity. Thus, this property is satisfied.

Property 3: Non-uniqueness

This property implies that there may be number of modules having the same complexity. TC abides this property, if the temporal cohesion of the modules is similar, and the complexity of the modules is also similar.

Property 4: Design details are important- (∃P) (∃Q) (P ≡ Q and |P| ≠ |Q|)

If two classes have the same functionality, they may differ in implementation. If the design implementation of two modules is different, TC produces different complexity values for each module.

Property 5: Monotonicity- (∃P) (∃Q) (|P|≤ |P; Q| & |Q| ≤ |P; Q|)

Let the two two modules P and Q be concatenated as P+Q. Hence, the complexity value of the combined class may be larger than the complexity of the individual classes P or Q. TC abides this property if there is a possibility of combining the modules P and Q and would share the attributes of the class while concatenation.

Property 6: Non-equivalence of interaction

The interaction between Example 1 and Example 2 are different which results in different metric value as shown. Hence, this property is also proved.

Property 7: Permutation

There are program bodies P and Q such that Q is formed by permuting the order of the statements of P and (|P| = |Q|). This property is not satisfied by the Object-oriented programs.

Property 8: Renaming

If module P is renamed as Q then |R| = |S|. This property requires that complexity of a module should not get affected by renaming it. TC does not have any impact over the change of name of module, hence TC satisfies property 8.

Property 9: Interaction increases complexity

The property says that the metric value for complexity of a class combined from two classes may be greater than the sum of two individual class complexity measures. This property is satisfied with TC as the complexity of the combined classes increases than the individual complexities. Summary of the TC validation is described in Table II.

Table II : TC values against Weyuker's Metric

Metric	P1	P2	P3	P4	P5	P6	P7	P8	P9
TC	Y	Y	Y	Y	Y	Y	N	Y	Y

VII. CONCLUSION & FUTURE WORK

The primary focus of this paper is to insist upon the invention of new software metrics based upon the most important quality factor of Software called as Cohesion. Temporal. cohesion metric is a software metric that is incorporated in the testing phase of a software development life cycle. It determines the percentage of Temporal cohesion that exists in a module or in a class. The software metric helps the developers to evaluate their Software programs so that the coding may be fine-tuned according to their need. The temporal cohesion satisfies eight out of nine properties of Weyuker's metric suite. Hence it is proven to be a qualified metric to be deployed in software industries so that quality products can be developed.

In future, the metric needs to be redefined as follows.

- In future, this work may be extended to invent some more metrics that could possibly identify the presence of all types of cohesion.
- An integrated approach is needed to include all cohesion measures in a single metric.
- Cognitive aspect of complexity need to be included to give more accurate measures.
- These metrics need to be evaluated with the real-time projects.

VIII. REFERENCES

[1] IEEE, "IEEE Standard Glossary of Software Engineering Terminology", IEEE Std. 610.12-1990. Institute of Electrical and Electronics Engineers, 1990.

[2] Paul Goodman, "Software Metrics Best Practices for Successful IT Management", Rothstein Associates Inc, 2004.

[3] Gui and Paul D. Scott. "Measuring software component reusability by coupling and cohesion metrics", Journal of computers 4.9 pp- 797-805, 2009.

[4] Mann, Ankita, Sandeep Dalal and Dhreej Chillar. "An effort to Improve Cohesion Metrics Using Inheritance", International Journal of computational Engineering Research (IJCER), 2013

[5] Amardeep kaur and Puneet Jai kaur, Class Cohesion metric in object Oriented Systems, international Journal of Software and web sciences.

[6] Desouky, Amir F and Letha H. Etzkorn. "object Oriented cohesion metrics: A Qualitative Emperical Analysis of Runtime Behavior." Proceeding of the 2014. ACM southeast Regional conference. ACM, 2014.

[7] Marcus, Andrian and Denys Poshyvanyk "The Conceptual cohesion of Classes", Software maintenance. 2005. Proceedings of the 21st IEEE international conference on IEEE,2005.

[8] S. Hari Ganesh and H.B. Vincent Raj, "A Novel Co-Functional Cohesion Complexity Metric: A quality based Analysis", International Journal of Applied Engineering Research (IJAER), Volume 10, Number 85, 2015

[9] S. Hari Ganesh and H.B. Vincent Raj, "A Theoretical Analysis SCOM: A Software Metric", International Journal of Control Theory and Applications (ISSN: 0974-5572), pp. 137-145, 2016

[10] S. Hari Ganesh and H.B. Vincent Raj, "CCOM – A Communicational Cohesion Metric for Object Oriented Programming", International Journal of Computer Applications (0975 – 8887) Volume 155 – No 5, December 2016.