ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)
Vol. 2, Issue 3 (July - Sept. 2014)
International Journal of Advanced Research in Computer Science & Technology (IJARCST 2014)

A Software Complexity Metrics for Modified Approach on LCOM

[I]S.Suriya, [II]F.Agnes Rosy
[I]Assistant Professor, [II]M.Phil Scholar
[I,II]Dept. of Computer Science, St.Joseph College Trichy, Tamilnadu, India

Abstract
Ensuring the quality in software designing and coding is highly recommended for enhancing the lifetime of software. There are many researches still going on to instill quality in coding. Software metric is considered as a tool to evaluate the software code that discloses the pros and cons involved in it. Though, there are plenty of metrics available in the software market, this paper is aimed to propose a new metric for software cohesion.

Keywords
Metric, Complexity, Cohesion, LCOM, Modified Approach- LCOM

I. Introduction

Over the past years, hardware was the major cost component of any information system. However, software has gradually become the larger part of the cost equation. Software metric is a standard way of measuring some attribute of the software development process or product. In simple words, software metric is a measurement. It measures such things as: product reliability, module size, defects, cost and time to develop, and design complexity. Evaluation has taken place everywhere in the real world right from assessing the students, the quality of product, selecting a candidate for a job and so on. Since evaluation has become a part parcel of our real life, it is inevitable to evaluate the software being built in this technical area.

II. Definitions of Software Metric

Software metric is a measure of some property of a piece of software or its specifications. Since quantitative measurements are indispensable in all scientific disciplines, there is a continuous attempt by computer science practitioners and theoreticians to bring similar approaches to software evolution. The goal is gaining objective, reproducible and quantifiable measurements, which may have frequent valuable applications in plan and budget preparation, price estimation, quality software debugging, assurance testing, software performance optimization, and optimal personnel task projects.

A. The Need for Software Metrics

The software crisis must be addressed and to the extent possible, resolve. To do so requires a more accurate schedule and total estimates, improved quality products, and higher productivity. All these can be accomplished through more effective software management, which, in turn, can be helped by the improved utilization of software metrics. The Current software management is ineffective because software development is exceedingly complex, and we have few well-defined, reliable measures either the procedure or the product to guide and measure growth. Thus, accurate and effective planning, estimating, and control is nearly possible to achieve. Improvement of the manage process depends upon the improved capability to identify, measure and controller essential parameters of the growth process. This is the goal of software metrics- the identification and measurement of the essential parameters that affect software development.

Software metrics and models have been proposed and used for some time. Metrics, however, have rarely been used in any regular, methodical fashion. Recent results show that the conscientious implementation and application of a software metrics program can support to achieve better management results, both in the short run and in the long run. Most software metrics cannot meaningfully be discussed in isolation from such metrics programs. Better use of previous metrics and development of improved metrics appears to be a significant element in the resolution of the software crisis.

B. Cohesion

In computer programming, cohesion refers to the level to which the components of a module belong together. Therefore, it is a standard of how strongly related each piece of functionality expressed by the source code of a software module is.

Coherence is an ordinal type of measurement and is commonly identified as “low cohesion” or “high cohesion”. Modules with high cohesion tend to be better because high cohesion is associated with several desirable qualities of software, including robustness, reusability, reliability and understand ability whereas low cohesion is associated with undesirable traits such as being difficult to maintain, hard to test, difficult to reuse, and even difficult to sympathize.

Cohesion is often contrasted with coupling is a different model. High cohesion often correlates with loose coupling, and vice versa. The objective of this paper is to find a new metric for cohesion. This paper is organized as follows. Survey on various Cohesion Metric is presented in section 2. New metric MALCOM is proposed in Section 3. Evaluation of the Metric is exhibited in Section 4 and Section 5 contains the conclusion of the paper.

III. Survey of Cohesion metrics

Briand et al. define a set of cohesion measures for object-based systems [2, 3] which are adapted in [4] to object-oriented systems. For this adapting a class is viewed as a collection of methods and data declarations. A data declaration is a local, private and public type declaration, the class itself or public attributes. There can be data declaration interactions between attributes, classes, types of different classes and methods.

They describe the following measures; Ratio of Cohesive Interactions (RCI), Neutral Ratio of Cohesive Interactions (NRCI), Pessimistic Ratio of Cohesive Interactions(PRCI) and Optimistic Ratio of Cohesive Interactions (ORCI).

Hitz and Montazeri base their cohesion measurements LCOM3, LCOM4 and C (Connectivity) on the work of Chidamber and Kemerer [5].The cohesion measurements by Bieman and Kang are also based on the work of Chidamber and Kemerer [6]. They define measurements known as Tight Class Cohesion(TCC) and Loose Class Cohesion (LCC). These metrics also consider pairs of methods which use common attributes; however a distinction

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

is made between methods which access attributes directly or indirectly. They also take inheritance into account, creative suggestions on how to arrange with inherited methods and inherited attributes. Lee et al. propose a set of cohesion measures based on the information flow through method invocations within a class [7]. For a method m implemented in a given class c, the cohesion of m is the number of requests to other methods implemented in class c, weighted by the number of limits of the invoked methods. The greater number of parameters an invoked method has, the more information is accepted, the stronger the link between the invoking and invoked method. The cohesion of a class is the sum of the cohesion of its methods. The cohesion of a set of classes is given by the sum of the cohesion of the classes in the set. Henderson-Sellers propose a cohesion measure (LCOM5) [8].
They state that a value of zero is obtained if each method of the class references every attribute of the class and they called this perfect cohesion". They also describe that if each method of the class references only a single attribute, the measure yields one and those values between zero and one are to be interpreted as percentages of the perfect value. They do not state how to deal with inherited methods and attributes.

IV. New Metric MALCOM

Brief overview of existing LCOM Metric
For a given class C with a number of methods, M1, M2, ...,Mn, let {Ii} be the set of instance variables accessed by the method Mi. As there are n methods there will be n such sets, one set per method. The earlier LCOM metric is then determined by counting the number of disjoint sets formed by the intersection of the n sets. However, this was found to be quite ambiguous and the pair later redefined their metric (LCOM2). For a class C1 with n methods, M1…..,Mn, let {Ii} be the set of instance variables referenced by method is Mi. There are no such sets I1……..,In.
We can define two disjoint sets:
P = {(II, Ij} |Ii∩Ij=∅}
Q = {(II, Ij} |Ii∩Ij≠∅}
The LCOM in methods is then described from the cardinality of these sets by:
LCOM = |P| - |Q|
if |P| > |Q| or 0 otherwise

A. MALCOM Metric

It is a new metric that overcomes the difficulties of LCOM metric. LCOM metric describes only the presence of Cohesion in a module. This makes an undesired conclusion on how good the class is? Another problem with LCOM is that metric does not get normalized or standard values. Our new proposed metric exhibit not only the presence of cohesion, but also the level of cohesion like Low, High and Medium.
For a given class C with a number of methods, M1, M2, ...,Mn, let {Ii} be the set of instance variables accessed by the method Mi. As there are n methods there will be n such sets, one set per method. The MALCOM metric is then determined by counting the number of sets which is not differs from zero by the union of the n sets. For a class C1 with n methods, M1…..,Mn, let {Ii} be the set of instance variables referenced by method Mi. There are number of n such sets I1 to In.
If (Number of Variables present in Two Methods-Number of Variables present after union ==0)
Increment R
Else Increment Q
Then do Q-R
If the result is positive it represents High Cohesion, Zero represent Equal levels of Cohesion and a negative value represent a low cohesion in the module.

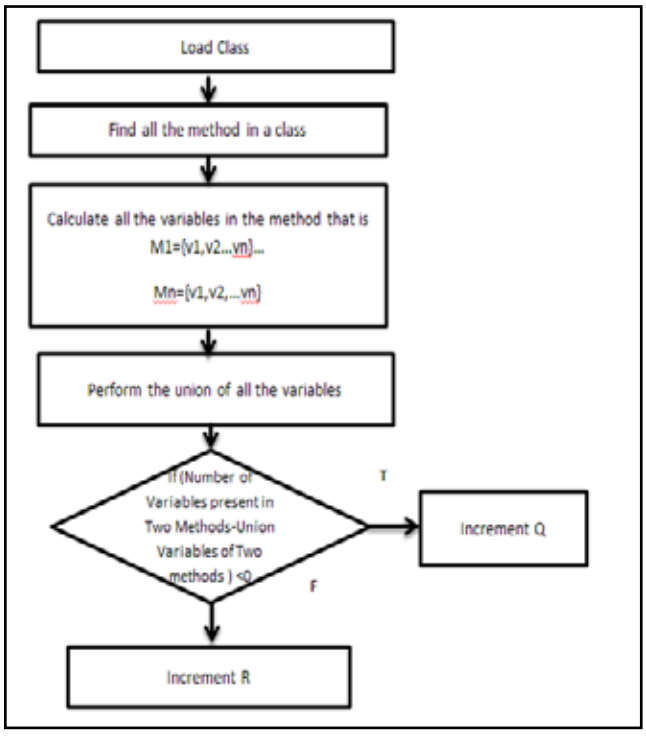
B. Example

Variables are: A, B, C, D
The functions are: M, N, O, P
Function M Contains variables {A, B, C}
Function N Contains variables {C}
Function O Contains variables {A, D}
Function P Contains variables {A, C, D}

C. Using MALCOM formula:

If (TNAV-TNUV) > 0INCREMENT Q otherwise INCREMENT P
Function M union N= {A, B, C} TNAV=4, TNUV=3 (4-3) = 1 Q=1
Function M union O= {A, B, C, D} TNAV=5, TNUV=4 (5-4) = 1 Q=2 (Increment Q)
Function M union P={A,B,C,D} TNAV=6, TNUV=4 (6-4) = 2 Q=3 (Increment Q)
Function N union O= {A, C, D} TNAV=3, TNUV=3 (3-3) = 0 R=1 (Increment P)
Function N union P={A,C,D} TNAV=4, TNUV=3 (5-4) = 1 Q=4 (Increment Q)
Function O union P={A,C,D} TNAV=4, TNUV=3 (5-4) = 1 Q=5 (Increment Q)
Total Number of methods that do not directly shares attributes whose value =1
Total Number of methods that directly share a common attribute =5
MALCOM=|Q|-|R|
(I e) 5-1=4

D. Architecture



V. Conclusion

A higher cohesive class is always preferable since they share all the variables within in the methods of the class in which they belong to. It means incrementing Q is always preferable. The most unwanted situation is incrementing R which signifies that the methods are not sharing any of the variables within themselves, even if they are belonging to the same class. In such situations the class has to be decomposed into two or more modules.

References

[1] *Mitchell K. D. Ram, " Total Quality Management in Software Engineering", IEEE, 1995Data Integrity, from Wikipedia, the free encyclopedia,[Online]http://en.wikipedia.org/wiki/Data_integrity.*

[2] *L.C. Briand, S. Morasca, and V. Basili. Measuring and assessing maintainability at the end of high-level design. In International Conference on Software Maintenance, pages 88 {97, Montreal, Canada, 1993.*

[3] *L.C. Briand, S. Morasca, and V. Basili. Defining and validating high-level design metrics. Technical Report CS-TR 3301, Department of Computer Science, University of Maryland, College Park, MD 20742, USA, 1994.*

[4] *L.C. Briand, J.W. Daly, and J.K. Wust. A unified framework for cohesion measurement in object-oriented systems. Empirical software Engineering: An International Journal, 3 (1): 65 {117, 1998.*

[5] *M. Hitz and B. Montazeri. Measuring coupling and cohesion in object-oriented systems. In International Symposium on Applied Corporate Computing, pages25 {27, Monterrey, Mexico, October 1995.*

[6] *J.M. Bieman and B.K. Kang. Cohesion and reuse in an object-oriented system. In ACM Symposium on Software Reusability, pages 295 {262, Seattle, Washington,USA, 1995.*

[7] *Y.S. Lee, B.S. Liang, S.F. Wu, and F.J. Wang. Measuring the coupling and cohesion of an object-oriented program based on information now. In International Conference on Software Quality, pages 81 {90, Maribor, Slovenia, 1995.*

[8] *B. Henderson-Sellers. Software Metrics. Prentice Hall, Hemel Hempstaed, U.K.,1996.*