International Journal of Engineering Research and Development
e-ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com
Volume 12, Issue 1 (January 2016), PP.74-79

A Review on Determining Cohesion and Coupling Based Object Oriented Metrics

S. P. Sreeja [1], Binoj M.[2], Kavitha S. N.[3]
[1,2,3] *Faculty, Department of Computer Applications, New Horizon College of Engineering, Bangalore-560 103, India.*

Abstract:- Object Oriented (OO) metrics play a key role in determining the efficiency of the code being developed under OO approaches. Cohesion and coupling are the widely used measures applied for the determination of important factors including reusability, maintainability and readability. There exists many Cohesion and Coupling based OO metrics making it difficult to choose the appropriate one. Moreover, improper application of these metrics may also lead to wrong judgements. This paper aims to provide a classification of various Cohesion and Coupling based OO metrics.

Keywords:- Object Oriented; Cohesion; Coupling; Classification of cohesion metrics; information related; Static, Dynamic..

I. INTRODUCTION

OO design and development are becoming very popular in today's software development environment. OO development need not only a specific approach to design and implementation, but also requires a unique way to measure the efficiency. Since OO technology uses objects as its fundamental building blocks, the methodology to measure the efficiency using software metrics for OO programs must be different from that of the conventional ones. OO software developments are handled differently from more traditional functional decomposition and data flow development methods. These are commenced by considering the system's behavior and/or data separately. OO analysis considers the problem by looking for system entities that combine them. OO analysis and design focuses on objects as the primary agents involved in a computation; each class of data and related operations are collected into a single system entity. Since there are various OO metrics available, it is essential to decide the type of metric that is best suited for the environment. Selection of metric is based on the specific needs of a software project that can be used for measuring the quality of software. ISO/IEC, a consortium for International Standard on software product quality states, "Internal metrics are of little value unless there is an evidence that they are related to external quality"[1]. The validity of these metrics needs to be checked thoroughly by using various ongoing projects at run time.

There exist few OO metrics that are most popular and used frequently. Lack of Cohesion in Methods (LCOM) is one of the prevalent metrics that is used to measure the dissimilarity of methods in a class by instance variable or attributes [2]. According to [3], Cohesion is defined as a measure of the degree to which the elements of a module belong together. There are various OO cohesion metrics that have been suggested by many researchers in the past several years. High cohesion always reduces the complexity of the modules or software. Even though there are various metrics available, it is a difficult task to determine the best suited cohesion metric.

Coupling is considered to be one of the common measures for determining OO metrics. Coupling Between Objects (CBO) is a count of the number of other classes to which a class is coupled [2]. When the methods or attributes of one class is used by another class it said to be coupled. Maintenance becomes a tedious process if the coupling is higher and also the changes in the class may affect the design stage. Even though similar classifications exist for coupling, cohesion metric is considered to be quite significant in [4]-[6]. Coupling refers to the degree of direct and indirect dependencies between parts of the design. To measure coupling in class diagrams there are various types of metrics [7]. A measure of coupling is more useful to determine the complexity. The higher the inter object coupling, the more rigorous the testing needs to be.

The significant contribution of this paper includes classification of OO cohesion metrics. Some of the similar surveys are quoted in section II. Section III gives the detailed information about the classification of OO cohesion metrics. The classification enlightens the researchers and academicians to proceed further with a specific area or branch that helps to focus on the various implications of it.

II. RELATED WORK

The authors have identified a few related research, where each one of them explores the various OO metric suites. The basic metric suite for OO design which provides a faster feedback for the designers and managers is introduced in [7]. The feedback enables to improve the quality of the software product during its development phase. Almost all the existing metrics are mentioned in the work [8] just to create an awareness of the existence of such metrics among the readers. Using meta-metrics the metrics mentioned by the authors are evaluated that acts as an aiding tool to select the appropriate one.

Comparison of various cohesion metrics are carried out in [9], to determine the best suited ones and the statistical measurement shows how to identify the same kind of cohesion. Cohesion metrics are compared with each other and the statistics for various cohesion measures are provided. To determine the variance, a principal component analysis is also carried out. Different tools were used for procuring all these measurements.

Fault-proneness prediction of OO classes is introduced in [10]. The authors presented an empirical validation of the OO metrics to predict the software quality. Three metric suites were taken into consideration where the similar components are identified along with the statistical models that determine the effectiveness in predicting the error-prone classes..

III. CLASSIFICATION OF OO COHESION AND COUPLING METRICS

Classification of the cohesion metrics varies based on the type and nature of the object that is used for developing particular software. In OO software development, classes play an important role. Especially, in cohesion, the quality of class design plays a vital role in deciding the cohesive factor. In several class cohesion metrics the categories for designing is based on the low level and high level designs. Low level design metrics are measured at the source code level and high level design are measured at the design stage. Development time, cost and quality can be improved by increasing the class cohesion.

A. Cohesion-Based Metrics

Cohesion is the degree at which methods within a class are related to one another and work together to provide well-bounded behaviour [11]. Different metrics have been developed based on the behaviour and similarity of the methods. High cohesion decreases the complexity and increases the software reliability [12]. The Metric suites surveyed are classified as Information and Non-information related based on the requisites of the software developed. Noninformation approach is to identify various cohesion metrics that are well suited for the object oriented environment. This leads to a division which is based on the design phase comprising of low-level and high-level designs. There are various cohesion metrics that are compatible on the applicability of the design stage. Several of these stages are properly understood by the researchers and there are many research carried out using the relationships of the software quality attributes. Reference in [13], have proposed a new set of measure that is based on the code level design that is based on information related. However, quite a few research work exist which is done during the design and implementation stages. A precise definition of coupling and cohesion together with an implementation approach through a tool named CCMETRICS is provided in [4]. The author provides a path to calculate metrics only from the source code.

1) Lack of Cohesion in Methods , Tight and Loose Class Cohesions (LCOM, TCC and LCC)

LCOM gives the degree of similarity that has to be measured which relies upon the class cohesiveness. Cohesiveness of the class is mainly depending upon how the different methods are utilized by the same set of instance variables to perform different operations. The LCOM value identified will help to indicate the dissimilarities, value zero for the class indicates that none of the methods in the class use any of the instance variables and thus there is no cohesiveness. Disparateness in the functionality provided by the class will have a high LCOM value.

Lot of work is carried out based on the extensions of LCOM, few of them are discussed here. High cohesion indicates the reusability and the simplicity of the classes are more and the low cohesion increases the complexity which leads to errors in the development process [2]. Reference [13] is a survey that has highlighted certain viewpoints that are related with LCOM. Measure for the attributes of an object in a class is based on the utilization of the instance variables and methods of a class. Classes can be split into subclasses when there is lack of cohesion. Inorder to promote encapsulation, identification of cohesion classes is essential.

An extended form of LCOM, namely Transitive LCOM, is proposed in [6]. The work is carried out on two open-source Java systems, where it addresses the transitive relation between class attributes and methods. An empirical study is carried out in which a logistic regression is applied for finding the fault proneness of the

system. The study enlightens the extended LCOM more accurately than the original (C&K) LCOM. The extended form considers transitive cohesion relation caused by method invocation.

Tight Class Cohesion (TCC) and Loose Class Cohesion (LCC) are the two cohesion measures that depend on the concept of direct and indirect connection of pairs defined in [3]. TCC measures the percentage of the relative number of directly connected methods and LCC measures the percentage of the relative number of directly or indirectly connected methods. These two measures of cohesion are similar to LCOM but, uses pairs of methods that shared common attributes. The usage of the attribute is calculated in different approaches are compared to its counterpart LCOM. Direct and indirect usage by the method of a class attribute is defined and has been differentiated. A method M uses an attribute directly if it appears as a data token. The method uses an attribute indirectly if the method M calls another method M', where M is a predecessor of M'. Using the transitive relations and with the help of a connected graph the direct and indirect methods are determined.

2) Low and High Level Designs

Low level design cohesion has been measured with the help of association and slice based approaches. This is used for measuring and supporting the software design, maintenance and restructuring. Two metrics are proposed based on the Input/Output Dependence Graph(IODG).

- Design Level Cohesion (DLC) - measured based on the level of a module determined by the relation level of output pairs
- Design level Functional Cohesion (DFC) - based on the dependence relationship of input/output components.

Association and slice based approaches are carried out in DLC and DFC respectively. Both analytical and empirical approaches are carried out to determine the comparison of the two cohesion measures. The conclusions arrived in [14] are based on the correlations used in IODG. Design-level measures are used for predicting cohesion in the code-level, since they are closely related with each other. Defects are identified with the help of the cohesion measures which can be further restructured.

Eventhough both the DFC and DLC are used for measuring the level of cohesion, according to [15], functional cohesion is the most desirable cohesion category. Functional cohesion is examined by using the data slice abstraction. The analysis done identifies the glue tokens, super-glue tokens and adhesiveness. Glue token, lies on more than one data slice. Super-glue token is used for identifying the data tokens that are common to every data slice in a procedure. And adhesiveness is based on the number of slices bounded with. These are useful in tracing the strong and weak functional cohesions in a procedure that is based on the relative number of super-glue tokens and glue tokens respectively. Rather than showing the reliability or maintainability in predicting the software attributes, the functional cohesion measure is derived to relate the attributes to one another. Low level designs can be measured during the design and implementation stages too and is discussed below.

High level design on the other hand measures the quality of the software at the designing phase. Few work identified by the authors are discussed here. The production of the better software relies upon the early development stages and the product produced involving all the measures will be a quality one. A class cohesion metric namely Distance Design-based Direct Class Cohesion (D3C2) metric, which used the Direct Attribute Type (DAT) matrix to measure the method to method interactions caused by sharing attribute types, attribute to attribute interactions caused by the attribute within the methods and attribute to method interactions is proposed in [16]. Further, a DAT matrix can be generated by using the data members and member functions of that particular class.

3) Refactoring and Program Slicing

Refactoring is done during the design phase of a cohesion metric that aims to improve the code maintainability and understandability. Fault proneness can easily be identified using the refactoring. High cohesive classes are less prone to faults. Few research has been carried out for predicting the errors. A novel metric called Lack of Coherence In Clients (LCIC) [17], is used for coherent set of roles in the program.

Calculation of LCIC is similar to that of LCOM, where the value ranges from 0 to 1. LCIC is calculated as the average of the ratios of the features related to the class through the client interface. Based on the statistical analysis, the LCIC has certain variations from the internal cohesion metrics. The variations included are: LCIC returns lower values compared to LCOM and LCIC is high when the class creates one object internally. The evaluation of the metric carried out is based on the design patterns to get better designs and it is

complex than the traditional OO code. Evaluation is not only carried out with design patterns but also it includes the refactoring evaluations to know the quality of the code.

To explore the impact of including or excluding special methods on cohesion measurements is attempted in [18]. An empirical study is carried out by using four different scenarios that shows how the refactoring and fault predictions are affected. The four different scenarios considered are by including all special methods, ignoring only constructors, ignoring only access methods and ignoring all special methods. The author has recommended for including the constructors in a class and excluding the access methods of a class for the refactoring activities. Similarly, the author also identified that the exclusion of constructors and inclusion of access methods were harmful for refactoring and negligibly effective when predicting the faulty classes.

4) Static and Dynamic Slicing

Static slicing does not make any assumptions regarding the input of a program. Both interprocedural and intraprocedural can be computed in static slicing. A new metric for interprocedural slicing is defined which is used for determining the estimation based on the size and position of the elements which are under process. A module is considered that needs to be sliced by removing the nonessential statements with the help of a criterion[19]. Mincoverage and maxcoverage are introduced to find the ratio of the shortest and largest slices respectively in the modules [20]. On the other hand, in [21] the authors have presented an algorithm for the denotational program slicer that is used for handling the functions and procedures.

Interprocedural slicing determines the rate of cohesion only within a single procedure. On the other hand, intraprocedural slicing deals with multiple procedures. Similarity based functional cohesion metric is found to measure the functional cohesion of a module in a procedural or OO program. For measuring the functional cohesion, the metric uses a data slice of the module as a basis. The modules are taken individually and it is concerned with intraprocedural static slicing. Six applications in the field of computer network were taken for the experimental analysis [22].

However, dynamic slicing assumes fixed input for a program and only the dependences that occur in the execution of the program are considered. Data and control dependences are taken into account for a specific program. There are various approaches used for dealing with the dynamic slicing to determine their accuracy and efficiency. The approaches that are dealt in [20] with dynamic slicing are basic algorithms, procedures, composite data types and pointers, concurrency and comparison. Program slicing includes various applications like debugging, testing and software maintenance where the accuracy and efficiency needs to be determined.

B. Coupling-Based Metrics

Coupling refers to the degree of direct and indirect dependencies between parts of the design. To measure coupling in class diagrams there are three types of metrics [23]. In this paper two CK metric is added to measure coupling performance. A measure of coupling is more useful to determine the complexity. The higher the inter object coupling, the more rigorous the testing needs to be. In this paper, three Genero metrics are used to validate the proposed approach check the performance.

1) Number of children metrics

Number of children metric defines number of sub-classes subordinated to a class in the class hierarchy. This metric measures how many sub-classes are going to inherit the methods of the parent class. Number of children metric relates to the notion of scope of properties. If Number of children metric grows it means reuse increases. On the other hand, as Number of children metric increases, the amount of testing will also increase because more children in a class indicate more responsibility. So, Number of children metric represents the effort required to test the class, reuse and maintain. The number of Association per class metric is defined as the total number of associations a class has with other classes or with itself. This metric is used to measure complexity and coupling [24]. When the number of associations are less the coupling between objects are reduced. This metric was introduced by Brian.

2) Coupling Between Number of Objects

A class is coupled with another if the methods of one class use the methods or attributes of the other class. An increase of Coupling between Number of Objects indicates the reusability of a class will decrease. Multiple accesses to the same class are counted as one access. Only method calls and variable references are counted. Thus, the CBO values for each class should be kept as low as possible [25].

3) Number of Dependencies In and Out

The Number of Dependencies In metric is defined as the number of classes that depend on a given class [10]. This metric is proposed to measure the class complexity due to dependency relationships. The greater the number of classes that depend on a given class, the greater the inter-class dependency and therefore the greater the design complexity of such a class. When the dependencies are reduced the class can function more independently. The Number of Dependencies Out metric is defined as the number of classes on which a given class depends. When the metric value is minimum the class can function independently.

IV. SUMMARY

Certain significance related to metrics such as LCOM, TLCOM, CLCOM, CBO and some of the drawbacks are also highlighted. On the other hand, metrics like TLCOM, CLCOM, Dependency In -Out take precautionary measures for improving the cohesion rate. The efficiency of the software can be controlled at the implementation time by adjusting certain measures.

V. CONCLUSIONS

This paper has attempted to consolidate the various Cohesion and Coupling based metrics from the available literature. Such a classification would hopefully help the OO designers to go for the appropriate choice in testing their code. However experience also plays a major role in the discrimination process. Our future work would involve analysing the suitability of OO metrics for the new age applications.

REFERENCES

[1]. Source:http://portal.acm.org

[2]. Linda H. Rosenberg, “Applying and Interpreting Object Oriented Metrics”, Presentation, Track 7: Measures/Metrics

[3]. James M Bieman, Byung-Kyoo Kang, "Cohesion and Reuse in an Object-Oriented System", ACM Symposium on software Reusability(SSR '95), April 1995, pp 259-262

[4]. Sukainah Husein, Alan Oxley, "A Coupling and Cohesion Metrics Suite for Object-Oriented Software", IEEE International Conference on Computer Technology and Development, 2009, pp 421-425.

[5]. Béla Újházi, Rudolf Ferenc, Denys Poshyvanyk and Tibor Gyimóthy, "New Conceptual Coupling and Cohesion Metrics for Object-Oriented Systems", IEEE Working Conference on Source Code Analysis and Manipulation, 2010, pp 33-42.

[6]. Jehad Al Dallal, "Transitive-based object-oriented lack-of-cohesion metric", Elsevier,WCIT 2010, Procedia Computer Science 3, 2011, pp 1581-1587.

[7]. Seyyed Mohsen Jamali, "Object Oriented Metrics-A Survey Approach", 2006.

[8]. M. Xenos, D. Stavrinoudis, K. Zikouli and D. Christodoulakis, "Object-oriented Metrics a Survey", Proceedings of the FESMA, 2000.

[9]. Letha H. Etzkorn, Sampson E. Gholstonb, Julie L. Fortune, Cara E. Stein, Dawn Utley,Phillip A. Farrington, Glenn W. Cox, "A comparison of cohesion metrics for object-oriented systems", Elsevier, Information and Software Technology 46, 2004, pp677–687.

[10]. Hector M. Olague, Letha H. Etzkorn, Sampson Gholston, and Stephen Quattlebaum, "Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes, IEEE Transactions on Software Engg., June 2007, Vol. 33, No. 6, pp 402-419.

[11]. Linda Rosenberg, Lawrence E Hyatt, "Software Qulity Metrics for Object-oriented Environments", Crosstalk Journal April 1997.

[12]. A.Yadav, R, A. Khan, "Class Cohesion Complexity Metric(C3M)", IEEE, International Conference on Computer & Communication Technology, 2011, pp 363-366.

[13]. Andrian Marcus, Denys Poshyvanyk, "The Conceptual Cohesion of Classes", IEEE Computer Society,Proceedings of 21st International Conference on Software Maintenance(ICSM '05), 2005, pp133-142.

[14]. James M. Bieman, B.K. Kang, "Measuring Design-level Cohesion", IEEE Transactions on Software Engg.,Feb 1998, Vol. 24, N0. 2, pp 111-124.

[15]. James M. Bieman, Linda M. Ott, "Measuring Functional Cohesion", IEEE Transactions on Software Engg., Vol. 20, N0. 8, Aug 1994, pp 644-657.

[16]. Jehad Al Dallal, "A Design-Based Cohesion Metric for Object-Oriented Classes", International Journal of Computer Science, Engineering and Technology, 2007, Vol 1 No. 3, pp 195-200.

[17]. Sami Mäkelä , Ville Leppänen, "Client-based cohesion metrics for Java programs", Elsevier, Science of Computer Programming 74, 2009, pp 355-378.
[18]. Jehad Al Dallal, "The impact of Accounting for special methods in the measurement of object-oriented class cohesion on refactoring and fault prediction activities", Elsevier, Journal of Systems and Softwares 85, 2012, pp 1042-1057.
[19]. Frank Tip, "A Survey of Program Slicing Techniques", Technical Report Cs-R9438, CWI, Amsterdam, Netherlands, 1994.
[20]. Linda M Ott and Jeffrey J Thuss, "Slice based metrics for estimating cohesion", IEEE Comp. Society, Proceedings of theFirst International Metrics Symposium USA, 1993, pp. 71–81.
[21]. Lahcen Ouarbya, Sebastian Danicic & Mohamed Daoudi, Mark Harman, Chris Fox, "A Denotational Interprocedural Program Slicer", IEEE, 9th Working Conference on Reverse Engineering, 2002, pp 181-189.
[22]. Jehad Al Dallal, "Software similarity-based functional cohesion metric", Institution of Engineering and Technology Softw., 2009, Vol. 3, Iss. 1, pp. 46–57.
[23]. S.R. Chidamber, C.F. Kemerer, "A metrics suite for object oriented design", IEEE Transactions on Software Engineering 1994, Vol. 20, No. 6, pp 476–493.
[24]. Kailash Patidar, RavindraKumar Gupta,Gajendra Singh Chandel, International Journal of Advanced Research in Computer Science and Software Engineering 3(3), March - 2013, pp. 517-521