International Journal of Engineering Research and Development
***e**-ISSN: 2278-067X, **p**-ISSN: 2278-800X, www.ijerd.com*
Volume 4, Issue 5 (October 2012), PP. 30-38

A Survey on Different Approaches of Determining Cohesion Based Object Oriented Metrics

S. P. Sreeja[1], Dr. R. Sridaran[2]
[1]Faculty, Department of Computer Applications, New Horizon College of Engineering, Bangalore-560 103, India. (Research Scholar, Bharathiar University, Coimbatore, India
[2]Dean, Faculty of Computer Applications, Marwadi Education Foundation's Group of Institutions, Rajkot, Gujarat, India.

Abstract:—Object Oriented (OO) metrics play a key role in determining the efficiency of the code being developed under OO approaches. Cohesion is one of the widely used measures applied for the determination of important factors including reusability, maintainability and readability. There exists many Cohesion 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 based OO metrics together with their significances and limitations. Hope that this would be beneficial for the developers and testers to make proper choice.

Keywords:—Object Oriented; Cohesion; Coupling; Classification of cohesion metrics; Information related, Noninformation related

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].

Cohesion metrics can be broadly classified into information related and noninformation related. Structured data and unstructured data are treated as the information related cohesion metrics. A class or algorithm is used for measuring the structured data. Static, semantic, internal and external cohesions are the various forms that are dealt with few of the work in [7],[8]. On the other hand, unstructured data can be classified as semantic and textual information which consists of only identifiers and comments [5][9].

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 [10]. 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 [11] 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 [12], 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 [13]. 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.

Eventhough there exist few work related to cohesion metrics, they have not been discussed under the broad variant of information versus noninformation based. This paper presents such a classification of cohesion metric suite with its significance. The survey discussed here gives a broad variant of the cohesion metrics which would benefit the researchers and designers.

III. CLASSIFICATION OF OO COHESION METRICS

Classification of the cohesion metrics varies based on the type and nature of the object that is used for developing particular software. The schematic diagram in Fig. 1 depicts the various classifications of the cohesion metrics and is discussed in the following section.

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 [14]. Different metrics have been developed based on the behaviour and similarity of the methods. High cohesion decreases the complexity and increases the software reliability [15]. The Metric suites surveyed are classified as Information and Non-information related based on the requisites of the software developed.

B. Information Related Cohesion Metrics

OO softwares are based on the design and development of classes. The given problem is decomposed to form a class and further the classes are defined in such a way that the complexity of the problems is very much reduced. The rate of cohesion will be depending on the complexity that exists in a software or module. A class with decreased complexity produces high cohesion [15]. The software contains both structured data and unstructured data. Several cohesion metric suites exist for the structured data that are of class-based and algorithm-based, whereas the unstructured data are composed of the identifiers and comments, that helps in maintaining the readability and understandability of the software.

Structured data is classified as one of the information related cohesion metrics. Structured data can be measured either based on a class or algorithm. Operations carried out in a class such as moving a method from a class to another, extracting methods from a class constitutes the structural information. The structured data mostly deals with various forms of static, semantic, internal and external cohesions.

- Static measurement is done by collecting the metrics directly from the source code.
- Semantic information such as comments and identifiers embedded in the source code are examined.
- Internal information relies in the class source file contents.
- External information deals with the design and implementation stages.

Some of the research work which deals with one or more combination of these cohesions are being discussed here. A study carried out in [16] that is restricted to the static, structural and internal information. The authors have presented a study with a large number of classes for the cohesion metrics. The study is classified based on the type and range of measurement of classes and it is varied based on the cohesion metrics. Structural information such as instance variable and reference methods are preferred for further analysis. As an alternate to statistical approach, it deals with the distribution of measurements that are more informative. Several methods were examined using the instance variables defined in a class. The instance variables contribute for measuring the cohesion rate. Almost all the classes selected for analysing the rate of cohesion had atleast one method where there are no instance variables. Even though the authors claim that the work is the first large-scale study of this kind, the variation in the metrics and the bimodal behaviour fails to identify the clarity of the useful values. The study is carried out using the commercial applications coded in java.

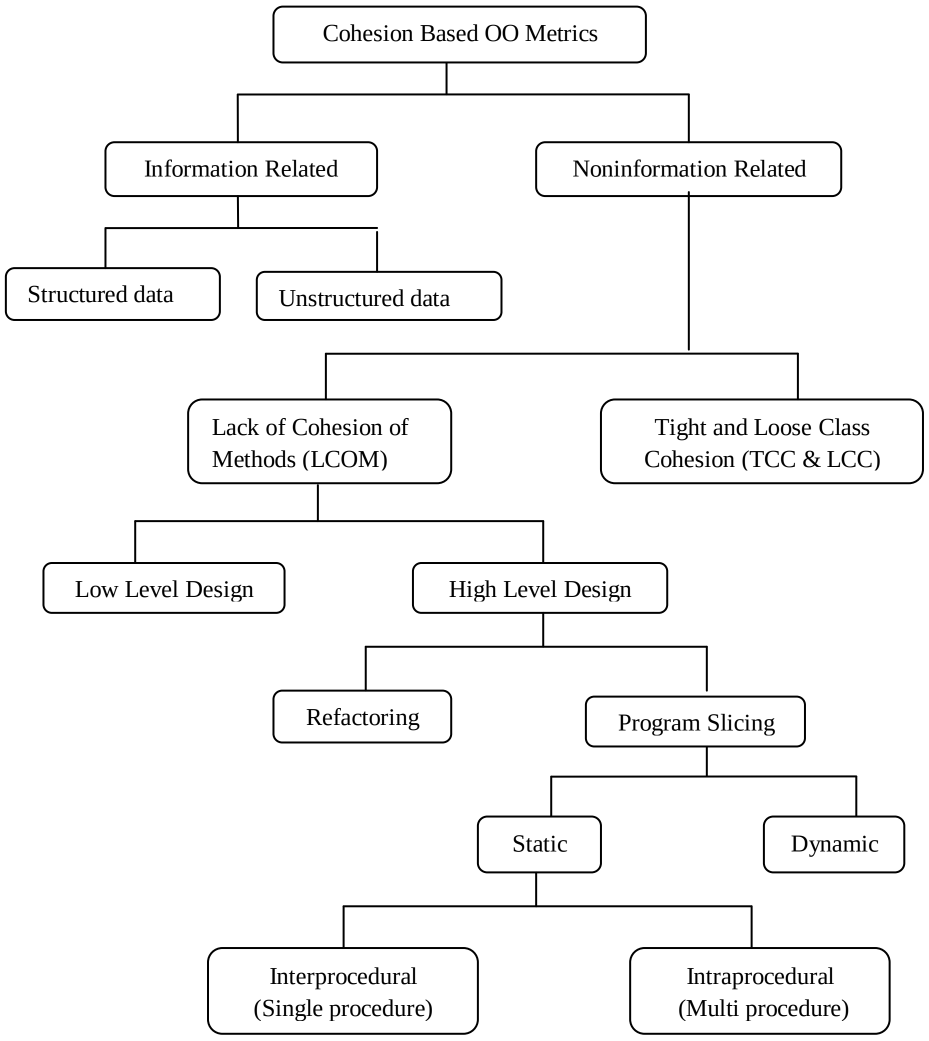
Cohesion Based OO Metrics
Information Related
Noninformation Related
Structured data
Unstructured data
Lack of Cohesion of Methods (LCOM)
Tight and Loose Class Cohesion (TCC & LCC)
Low Level Design
High Level Design
Refactoring
Program Slicing
Static
Dynamic
Interprocedural (Single procedure)
Intraprocedural (Multi procedure)

Fig. 1: Classification of OO cohesion metrics

On the other hand a combination of structural and semantic cohesion is presented in [17]. The extract class refactoring designed in [18] is considered for improving the class cohesion that aims at the conceptual similarity between the methods. The authors deal with the extraction of methods from low cohesion classes with a high structural and semantic similarity and further these methods are used for creation of new high cohesive classes. This is carried out by using the MaxFlow-MinCut algorithm [18], with the help of a directed graph with simple approaches. The performance measurements are analysed using the threshold values of the methods that are summarized below:

- Threshold used for the weight of the graph edges with a low value
- Structural and semantic information that should be used with a lesser value
- Performance improvement is made by combining semantic and structural information.

Papers discussed above are dealt with different combinations of structural information, however in [8], the authors have focussed only on structural information. The authors introduced a new class cohesion criterion that is based on interaction between class methods. Work carried out is by using the method invocation with two criterions: attribute usage and method invocation. This helps to capture the characteristics of classes and to measure the cohesion property. A cohesion measurement tool for Java program is developed and a case study is also performed. The proposed metrics captured more connected pairs, specifically the classes that implicitly take the interactions between methods.

The structured information can further be analyzed in detail through the class-based (internal view) or algorithm-based (external view). Reference in [19] has proposed in forming cohesion graphs between the clients of a class and its member variables which will be based on the external usage of class properties. External view on Lack of Cohesion in Methods (ELCOM) is defined based on the client classes where the class uses all its instance variables including the inherited ones. The value of ELCOM analyses whether the class is refined or unrefined in the design and implementation stages. The ELCOM is calculated by converting the class as a connected graph and if it is identified as loosely-connected, it reflects that the refactoring is essential for the class. Quite a few works exist which are class-based and algorithm-based [20].

Structured data can be of various forms with either one or more combinations. On the other hand, unstructured data can be classified as the semantic information which consists of only identifiers and comments. Conceptual Cohesion of

Classes (C3), measures the semantic information embedded in the source code [9]. This approach is by using the same type of information and is based on a similar interpretation of cohesion. Comments and identifiers from the methods are extracted and then processed. A new set of measures of individual classes have been provided for the usage of semantic information. Detailed analysis is carried out using open source software. The differences and similarities are found by using only theoretical evaluations. The methods used depend upon implemented classes or overloaded classes. C3 does not perform any distinction between constructors, accessors and other method stereotypes.

The extended form of C3 used in [9] is redefined in [17], analyses the unstructured information (comments and identifiers) that exists in the source code. New measures have been proposed that are used for predicting the software failures. The textual information has been extracted, represented and analyzed by using an Information Retrieval technique, called Latent Semantic Indexing (LSI). Conceptual relation of the methods to a class is identified. The results of the analysis are examined for evaluating the logistic regression models and certain quantitative characteristics such as precision, correctness and completeness are utilized. The authors carried out an analysis in three open-source software systems and the software faults are predicted and measured.

Eventhough the metrics discussed above give the details of the unstructured information based only on cohesion, two novel metrics are proposed in [5] that computes in a simpler way for both cohesion and coupling. The two metrics are Conceptual Coupling Between Objects (CCBO) and Conceptual Lack of Cohesion in Methods (CLCOM). An empirical study is dealt in with these two metrics, enabling to predict the bugs as similar to the structural metrics available in Columbus source code and it can improve the bug prediction. An extensive study using the machine learning techniques indicates that the measures are accurate. However, it can be further extended by applying advanced source code which consists of the textual information that needs to be expanded and split for predicting the bugs. Cohesion metrics can also be measured based on the noninformation related metrics, which is explained with few variations in the following section.

C. NonInformation Related Cohesion Metrics

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.

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

According to Chidamber and Kemerer (C & K) metric suite in [21], Cohesion is referred to as the internal consistency within parts of the design. 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 [9] 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.

There are various other classifications related to the connectivity between the methods that are based on the noninformation cohesion. Class abstraction determines the connectivity between the methods. Tight Class Cohesion (TCC) and Loose Class Cohesion (LCC) are the two cohesion measures that depends 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. According to the study, the authors have concluded the following:

- TCC is always less than or equal to the value of LCC.
- TCC and LCC indicates the visible methods and their degree of connectivity in a class,
- TCC and LCC measures the cohesiveness and also implies that increased level of reusability and inheritance of classes may lead to poor cohesion.

2) Low and High Level Designs

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.

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 [22] 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 [23], 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.

Reference in [9], 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. On the other hand, cohesion graphs formation between the clients of a class and its member variable that deals with the code based (Low-level) is proposed in [7]. To identify effectiveness of the classes during the design and implementation stage, ELCOM metric is used. The need for refactoring is identified based on the connectivity of the graph. The ELCOM is calculated by converting the class as a connected graph. If the graph is identified as loosely-connected, it reflects that the refactoring is essential for the class.

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 [24]. The DAT matrix is a k x l binary matrix, where k denotes number of methods and l, the number of distinct attribute type in the class. UML designs can be used for high-level design artefacts. D3C2 Metric is not defined, if there exist no methods/attributes in a class. To define a problem a UML class diagram can be drawn that incorporates all the data members and member functions. Further, a DAT matrix can be generated by using the data members and member functions of that particular class. The UML class diagram and DAT matrix for AccountDialog class used for measuring the metric is as shown in the below Fig. 2 and 3 [24].

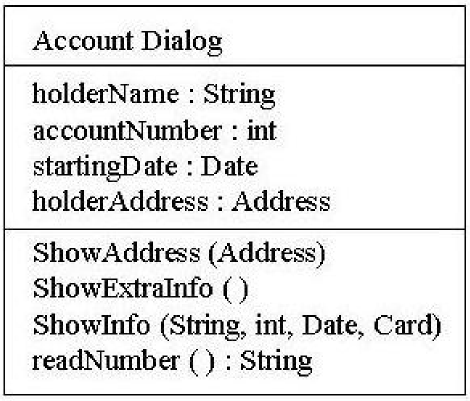

Fig. 2: UML class diagram for AccountDialog class[24]

	String	Int	Date	Address
showInfo	1	1	1	0
showAddress	0	0	0	1
showExtraInfo	0	0	0	0
readName	1	0	0	0

***Fig. 3:* The DAT matrix for the *AccountDialog* class[24]**

The detailed description is not available at the design level; however the information regarding class, attributes and method signature is available. By utilizing all such information the cohesion metrics can be developed. The two such metrics are the Normalized Hamming Distance (NHD) and Normalized Hamming Distance Modified (NHDM). NHD is a cohesion metric applicable to design level and the metric uses the parameter type of each method with an average agreement between the pair of methods. A Parameter-Occurence (PO) matrix is used, where the row represents the methods in a class and column indicates datatype which occurs as the type of parameter in the methods of the class [25]. However, NHDM is introduced in [26] which uses the similarity of parameter types of methods in a class. NHDM is a modified version of NHD suggested in [25] and attempted to overcome the anamoly of NHD. The NHDM metric identifies both the maximum and minimum range, based on the PO matrix by including the self parameter. This metric does not depend on the size of the class compared to its counterpart. Similarly, it gives different results for classes with different properties as against NHD. High level designs are measured not only based on signatures it also uses the interface relationship which is highlighted below.

According to [27], high level design of a software system is defined as a collection of module and subroutine interfaces related to each other by means of USES and IS_COMPONENT_OF relationships. Precise and formalized information on module or subroutine bodies is not available at this stage. The authors have presented a comparative study of various high level measures. The study carried out deals with few attributes of object-based design to find the fault proneness. Theoretical and empirical validation has been done on three real-life applications in Ada language at NASA, which were of property-based approach. The study is concluded with a statistical measurement using the abstract data types that indicate a good significance for a high level design. Design also involves the class and its behaviour for measuring the performance.

Static and dynamic cohesion are the two such variances of the design stage that helps to measure the OO software. Static cohesion is mostly based on the class and dynamic cohesion is based on the class behaviour. A set of metrics is proposed in [28] that is used for measuring the static and dynamic message flow. The flow between the objects member function in the detailed level, are used by various class relationships. These metrics are used for obtaining high performance in distributed applications. The authors have proposed for both static and dynamic cohesion metrics. Static cohesion metric finds out how often a method in a class invokes another class. Cohesion becomes smaller if the reference to a class by a method in another class is invoked frequently, but if it is invoked rarely then the cohesion becomes larger.

Dynamic cohesion metrics are measured depending on the number of messages and the loadsize of each message, if the loadsize is small the dynamic cohesion is also small. On the otherhand, a dynamic cohesion metric upto the object level by using the important features such as inheritance, polymorphism and dynamic binding is dealt in [29]. An experiment is carried out to show the accurateness by using 20 java programs and also it is validated empirically using Java Development Kit (JDK). The authors deal with the dynamic cohesion at run time by considering the objects read/write dependency. Attributes and methods determine the read dependence whereas the write dependence relies on the call or dependence between methods and attributes. The metric is an accurate one since it is highly influenced at run-time. Measurements are dealt with the object level rather than at the class level. Correlation is used for selecting the weight values through a common approach.

The design stage is considered to be the crucial stage of any software system. The complexity increases rapidly when the software size increases. Measuring the quality of the software plays a vital role in the design and development stage of any software system. As the system gets complicated some mechanism should be used for resolving the conflicts with the help of program slicing, refactoring, etc. Program slicing is defined as the portion of the program that might affect the value of a particular identifier at a specified point in the program [22]. Refactoring is defined as the process of changing an existing OO software code to enhance its internal structure while preserving its external behaviour [18]. The subsequent section explains the metrics used for resolving the conflicts occurred in the design stage.

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) [19], 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 [30]. 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.

High level design cohesion metrics are determined during the design stage, where program slicing is considered to be one of the measurements. A program slice consists of the parts of a program that affects the values at some point of time [31]. Program slicing is the task of estimating such program slices. The important classification is that of the static and dynamic slicing. Static slicing is calculated without making any assumptions on the program's input, whereas dynamic slicing depends upon the test cases used in a program. Slicing has many applications that include software maintenance, debugging, testing etc. The following section discusses some of the work carried out in static and dynamic slicing.

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[31]. Mincoverage and maxcoverage are introduced to find the ratio of the shortest and largest slices respectively in the modules [32]. On the other hand, in [33] 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 [34].

However, 13 program slicing metrics are introduced in [35] that are used for measuring the size, complexity and cohesion properties of programs. The program slicing metrics are used in C programs based on intraprocedural slicing whose output variables in every function are computed. The dependence graphs are used to represent the program slices. The authors developed a set of metrics at the function level and file level. Normalization is implemented for complex statements within the functions. In static slicing there are various basic methods used to compute the slices. The technique used for addressing the approaches includes algorithms, procedures, unstructured control flow, composite data types and pointers, concurrency and comparison. The algorithms and procedures can be represented based on dataflow equations, information flow and program dependence graph[31].

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. Similar to that of static slicing, dynamic slicing also consists of both the interprocedural and intraprocedural slicing. There are various approaches used for dealing with the dynamic slicing to determine their accuracy and efficiency. The approaches that are dealt in [32] 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.

Table I below provides the consolidated view of significance, drawback and tools used for various cohesion metrics.

IV. SUMMARY

Certain significance related to metrics such as LCOM, TLCOM, CLCOM are tabulated. (Refer Table 1). Some of the drawbacks are also highlighted together with different procedures to overcome. On the other hand, metrics like TLCOM, CLCOM take precautionary measures for improving the cohesion rate. The efficiency of the software can be controlled at the implementation time by adjusting certain measures as found in the table.

V. CONCLUSION AND FUTURE WORK

This paper has attempted to consolidate the various Cohesion 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 analyzing the suitability of OO metrics for the new age applications.

Table I: Consolidated view of Cohesion metrics

Metric	Classification	Significance	Drawback/ Limitations	Tools used
LCOM	Non Information based	LCOM value provides a measure of the relative disparate nature of methods in the class.	Possibility of increased complexity due to higher interaction between the classes.	——
TLCOM	Non information based	Addresses transitive relation between class attributes and methods. More accurate than LCOM according to the analysis.	Dealt only with LCOM and not for any other cohesion metrics.	——
CLCOM	Information based unstructured data	To predict the bugs in the source code.	Only textual information is considered.	——
ELCOM	Information based structured data	Tried to complement the internal cohesion with external cohesion view.	Does not provide accurate results due to nonelimination of constructors.	Tool-10KLOC (KiloLines of Code), Open source code such as Robocode, Art

				of illusion, jEdit
TCC & LCC	Non information based	Method of the class attribute is defined using the direct and indirect connected methods.	Dealt only with a single software, but not used in multiple softwares.	——
D_3C_2	High level design based	Uses the UML for high-level design based for direct class cohesion.	Indirect interactions and method invocations are not dealt with for measuring the metrics.	——
LCIC	High level design based Refactoring	Metric is used in high-level design to measure the coherence and determine the need for refactoring.	Supports only java programs.	Tool for java projects in eclipse environment
NHDM	High level design based	Anomalies of NHD are overcome. The metric does not depend on the size of the class.	——	Tool CohMetric for java based charting library

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]. Sami Makela, Ville Leppanen, "Client based Object-Oriented Cohesion Metrics", IEEEComputer Society, Proceedings on 31st Annual International Computer Software and Applications Conference(COMPSAC 2007), Vol 2, pp743-748.

[8]. L. Badri and M. Badri, "A Proposal of a New Class Cohesion Criterion: An Empirical Study", Journal of Object Technology, Vol. 3, No. 4, April 2004, Special issue:TOOLS USA 2003, pp 145-158.

[9]. 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.

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

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

[12]. 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.

[13]. 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.

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

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

[16]. Richard Barker, Ewan Tempero, "A Large-Scale Empirical Comparison of Object-Oriented Cohesion Metrics", IEEE Computer Society, 14th Asia-Pacific Software Engineering Conference, 2007, pp 414-421.

[17]. [17] Andrian Marcus, Denys Poshyvanyk, Rudolf Ferenc, "Using the Conceptual Cohesion of Classes for Fault Prediction in Object-Oriented Systems", IEEE Transactions on Software Engg., Mar/Apr 2008, Vol. 34, N0. 2, pp 287-300.

[18]. M. Fowler, "*Refactoring: Improving the Design of Existing Code*", Addison Wesley, 1999.

[19]. Sami Mäkelä , Ville Leppänen, "Client-based cohesion metrics for Java programs", Elsevier, Science of Computer Programming 74, 2009, pp 355-378.

[20]. Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, Clifford Stein, "*Introduction to Algorithms*", MIT Press 2001, 2nd Edition.

[21]. 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.

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

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

[24]. 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.

[25]. S. Counsell, S. Swift and J. Crampton, "The Interpretation and Utility of Three Cohesion Metrics for Object-Oriented Design", ACM Transactions on Software Engi-neering and Methodology, Vol. 15, No. 2, 2006, pp. 123-149.

[26]. Kuljit Kaur, Hardeep Singh, "Exploring Design Level Class Cohesion Metrics", Journal of Software Engineering & Applications, 2010, Vol. 3, pp 384-390.

[27]. Lionel C. Briand, Sandro Morasca and Victor R. Basili, " Defining and Validating Measures for Object-Based High-Level Design Metrics", IEEE Transactions on Software Engg., Vol. 25, N0. 5, Sep/Oct 1999, pp 722-743.

[28]. Eun Sook Cho, Chul Jin Kim, Soo Dong Kim, Sung Yul Rhew, "Static and Dynamic Metrics for Effective Object Clustering", IEEE, 5th Asia Pacific Software Engineering Conference (APSEC 98), Taiwan 1998, pp 78-85.

[29]. Varun Gupta, Jitender Kumar Chhabra, "Dynamic cohesion measures for object-oriented software", Elsevier, Journal of Systems Architecture, 57, 2011, pp 452–462.

[30]. 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.

[31]. Frank Tip, "A Survey of Program Slicing Techniques", Technical Report Cs-R9438, CWI, Amsterdam, Netherlands, 1994.

[32]. 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.

[33]. 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.

[34]. Jehad Al Dallal, "Software similarity-based functional cohesion metric", Institution of Engineering and Technology Softw., 2009, Vol. 3, Iss. 1, pp. 46–57.

[35]. Kai Pan, Sunghun Kim, E. James Whitehead, "Bug Classification Using Program Slicing Metrics", IEEE Comp. Society, Proceedings of the Sixth IEEE International Workshop on Source Code Analysis and Manipulation (SCAM'06), 2006.