A Retrospective Study on Cohesion and Coupling Metrics of OO Software Systems

[1]Manvi Khatri, [2]Dr.Puneet Goswami
[1]M.Tech Scholar, [2]Professor and Head
Department of CSE
SRM University, Haryana, India

***ABSTRACT*- The development of software is vigorous and always enduring the key changes. The prominence of OO design metrics is important for assessing the software complexity, cohesion, coupling, estimating size and quality. There are lots of frameworks for gauging the cohesion and coupling metrics in Object oriented design. For maintaining the high quality of software, developers always try to choose loosely coupled and highly cohesive design system. The purpose of this research study is to fortify the existing coupling and cohesion metrics used in OO environment by analyzing their significance, their proposed metric, divulging their benefits and drawbacks, and some of its future scope.**

***KEYWORDS* - Object-oriented, Cohesion, Coupling, Software, Metrics.**

I. INTRODUCTION

Today's software development industry gives their priority more towards the quality of software. So, it increases the majority of work accomplished for evaluating and predicting the quality of software. Object-oriented evaluation has become a popular research area. Object- oriented software is significantly distinct from software developed using careful and reliable methods. Object-oriented processes are recommended as the best way for interpretation of the software development problems. The object-oriented design metrics is indispensable in software engineering for quantifying the software complexity, estimating the quality, size and project efforts.

Development of object-oriented not only need a different approach for design and implementation, even also needs a different approach to software metrics. Software metrics helps us to verify and evaluate various facet of the complexity of a software product. Cohesion and Coupling weighed up to be the most important attributes. Coupling and cohesion are the attributes which are used to estimate the degree of interactions and relationships between elements of the source code like classes, methods or attributes in object-oriented systems. Lots of object-oriented metrics of Coupling and Cohesion have been developed as object-oriented analysis and design is viewed to be at the forefront of software engineering technologies. Developers always choose low coupled and highly cohesive design for maintaining high quality object-oriented software.

In object-oriented software systems, the classes replace modules with methods and attributes as their elements [Briand et. al. 1997]. In a summate, the object-oriented methodology essence on objects. The leading objectives for OO metrics are same as those for traditional software metrics [Rachel et. al. 1998] -

- To enhance the understanding of a product quality.
- To evaluate the effectiveness of the process.
- To reform the quality of work performed at a project level.

Static and Runtime metrics

To calculate the quality of the OO design lots of metrics has been proposed. Broadly, it is classified into two categories: 1) Static 2) Runtime/Dynamic Metrics.

The static metrics is used to gauge what will happen when the execution of a program takes place and measures the quantity and complexity of dissimilar features of the source code. Whereas, the dynamic metrics is used to quantify actually what will happen when execution of a program takes place and also come across the runtime behavior, complexity and characteristics of the source code. In between both the metrics, lesser work is done on dynamic metrics as compared to static ones as the analysis of runtime metrics is more expensive and very complex in terms of performance.

Coupling Measures - Coupling in OO software system uses to confine the strength to which each program module count on each one of the other modules.

- Static Coupling Metrics - There exists a large range for coupling quantification. The most widespread one is C-K metrics (CBO, RFC).
- Dynamic Coupling Metrics - A little research work has been done towards this field. Some metrics is given based on below dimensions- [Erik Arisholm et. al. 2004]
  - Mapping(object or class)
  - Direction (import or export)
  - Strength (number of dynamic parameters, distinct methods, distinct classes)

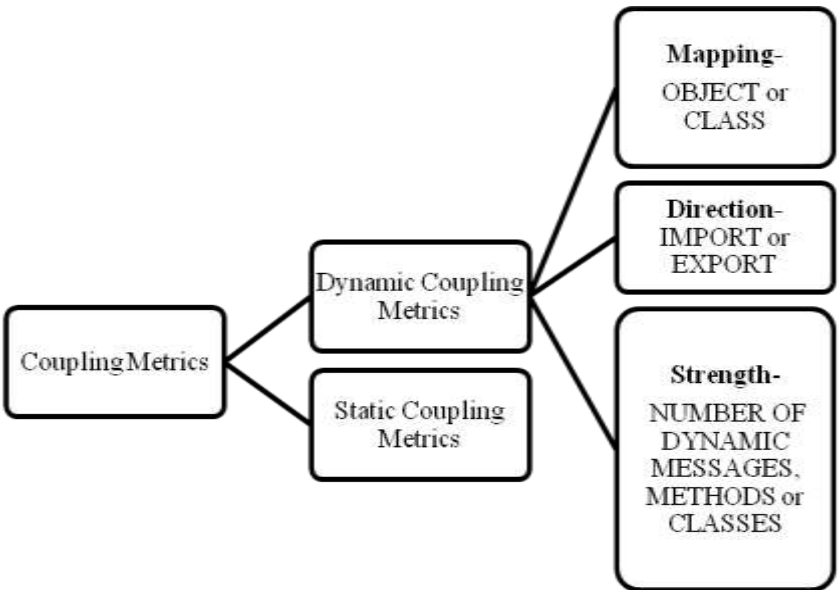

Fig. 1.1 Coupling metrics

Cohesion Measures - Cohesion in OO software system is the property that is used to enumerate how tightly bound does the elements of modules are.

- Static Cohesion Metrics - Many choices are present for measuring the cohesion. The most common are LCOM(C-K metrics).
- Dynamic Cohesion Metrics - Few metrics have been proposed for measuring cohesion at runtime.

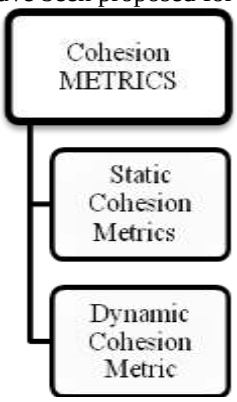

Fig. 1.2 Cohesion metrics

II. AN EVALUATION OUTLINE ACQUISITION

We have studied many proposed cohesion and coupling metrics. Each metric scrutinize is divided into the following segment:

Table 1.1

Year	Time of publishing
Objective	Summarize the goal or motivation behind metric.
Domain of Measure	Refers to the level of granularity- Fine grain, Medium grain and Coarse grain.
Proposed Metric Used	Describes the new projected measure/metric.
Applicable/Used On	Describes where the metric has been used on.
Description	Explanation about the proposed study.

UML	Unified Modeling Language, expected to use for measuring the cohesion and coupling metrics.
Validation	Determines how the metrics have been validated- Theoretically, Empirically, Experimentally, Heuristically.
Advantages and Disadvantages	Describes the benefits and shortcomings of the metrics.
Future Scope	Explain about the upcoming works to be done.

III. LITERATURE REVIEW

- **S.R. Chidamber and C.F. Kemerer**

Year: 1994
Objective: The main objective of this paper was to have expansion and empirical validation of a set of theoretically grounded metrics of OO design.
Proposed Metric Used:
Consider a class C_1 with 'n' methods – $M_1, M_2 \dots M_n$
$\{I_j\}$ = set of instance variables used by M_i methods
There are 'n' such sets - $\{I_1\} \dots \{I_n\}$
Let $P=\{(I_i, I_j) \mid I_i \cap I_j = 0\}$
$Q = \{(I_i, I_j) \mid I_i \cap I_j \neq 0\}$
And if all 'n' sets $\{I_1\} \dots \{I_n\}$ are 0 then let $P=0$

$$LCOM = \begin{cases} |P| - |Q|, & If\ |P| > |Q| \\ 0, & Otherwise. \end{cases}$$

CBO for a class = count of the number of other classes to which it is coupled.

$RFC = |RS|$
Where RS is the response set for the class.
And, $RS = \{M\} \cup_{all\ i} \{R_i\}$
Where, $\{M\}$ = set of the methods in the class
$\{R_i\}$ = set of methods by method 'i'
Used on: Large software systems
Description: The Chidamber and Kemerer metrics are also symbolized as "C-K Metrics". They proposed a set of six metrics for OO design. They were the ones who evaluated the research empirically for the first time. Out of six, CBO and RFC are the proposed coupling metrics and LCOM is the proposed cohesion metrics. The CBO metric is related to an object which is coupled with another object in which one acts on the other. If CBO values are high then it shows poor managerially results. The LCOM metric is used to count the number of method pairs whose similarity is zero minus the method pairs whose similarity is not zero. The high LCOM values result in lots of rework effort required. The RFC metric is used to define the number of methods which are executed in the response to a message received by the object of a class. If the value of RFC is more than testing becomes more complex and results in fault occurrence.
Validation: Both Theoretically (analytical) and Empirically.
Advantages and Disadvantages: This study was the first empirically validated plan for formal metrics of OOD.
It was noticed that somewhere C-K metrics had theoretic anomalies.
Future Scope: To chase the commercial applications from outset to development and congregate metrics at varied intermediate stages of the project. To evaluate the extent to which these metrics correlate with managerial performance indicators such as design, test, maintenance effort, quality and system performances.

- **Fernando Brito e Abreu, Rogério Carapuça**

Year: 1994
Objective: The objective of this study was to put forward a set of metrics for accessing the use of the main generalization of the OO paradigm such as inheritance, information hiding and gives focus on reuse, which helps in increasing the software quality and development productivity.

Proposed Metric Used:
Let, class C_C = Client of class C_S

C_S = Supplier of class C_C (whenever C_C contains at least one reference to a method or attribute of the class C_S)
This client-supplier relation by C_C will be –

$$is\_client(C_C, C_S) = \begin{cases} 1, \text{iff } C_C \rightarrow C_S \wedge C_C \neq C_S \\ 0, \text{otherwise} \end{cases}$$

$$COF = \frac{\sum_{j=1}^{TC}\left[\sum_{j=1}^{TC} is\_client(C_i, C_j)\right]}{TC^2 - TC}$$

Where $TC^2 - TC$ = Maximum value of client-supplier relations.
Used On: Real time projects
Description: The MOOD was designed which includes a set of metrics. The main perception of this model is an inheritance, encapsulation, information hiding or polymorphism, which are input to further reuse and easy maintainability. Coupling factor was one of the metrics in it. In this metric, as the number of client-supplier relations becomes larger, the complexity increases, the encapsulation, and potential reuse decreases and it also limits the understandability and maintainability.
Validation: Heuristically
Advantages and Disadvantages: The proposed metric has helped in setting OO design standards at the organization level and helped the OO specialists to direct their development process.
The drawback of this study was the values declared were inappropriate.
Future Scope: One of the major challenges in the future will be to combine the MOOD metrics except reuse factor to achieve generic OO software system complexity metric. They will plan to develop an Eiffel in the future as C++ stub was under construction. This tool can help in wide validation of the available systems and can obtain refined values for limits referred in the design proposal. Also, it will study the statistical independence of each MOOD metric towards each other.

- **James M. Bieman and Byung-Kyoo Kang**

Year: 1995
Objective: To assess the relationship between class cohesion and private reuse in the system. And build up the cohesion measures and relate them to a large C++ system.
Proposed Metric Used:

$$TCC(C) = \frac{NDC(C)}{NP(C)}$$

$$LCC(C) = \frac{NDC(C) + NIC(C)}{NP(C)}$$

Where
NP(C) =Total number of pairs of abstracted methods in a class AC(C)
NP = Maximum possible number of direct or indirect connections in a class
NDC(C)=Number of connections in AC(C)
NIC(C)=Number of indirect connections in AC(C)
Used On: Large C++ systems
Description: The two proposed measures of class cohesion are based on the direct and indirect connections of method pairs. The TCC is the relative number of the directly connected methods whereas the LCC is the relative number of the directly or indirectly connected methods. The value of LCC is always greater than or equal to the value of TCC. They are used to signify the degree of the connectivity between visible methods in a class. The visible methods are defined within the class or inherited to the class.
Validation: Experimentally (validated on real time system- Interview System, Stanford University)
Advantages and Disadvantages: The proposed class cohesion measures can be used to locate the classes which have been designed inappropriately.
The results show that the classes which have been heavily reused by using inheritance exhibit lower cohesion.
Future Scope: The results can be confirmed by the studies of additional software systems.

- **Pham Thi Quynh, Huynh Quyet Thang**

Year: 2009
Objective: The objective of this paper was to propose a suite of dynamic coupling metrics of service- oriented software.
Proposed Metric Used:

$$CBS = \sum_{i \neq j=1\ldots n}^{n} A_i B_j$$

Where,

n= number of services in system

$$A_i B_j = \begin{cases} 0, & A_i \text{ does not connect to } B_j \\ 1, & A_i \text{ connects to } B_j \end{cases}$$

$$IMS = \frac{fan.out}{fan.in + fan.out} \times 100\%$$

$$IMS = \begin{cases} 0, & \text{stability of service is very high} \\ 1, & \text{stability of service is very unstable} \end{cases}$$

Where,
fan.in metric means the number of the messages which are sent to service A
fan.out metric means the number of messages which are sent by service A

$$DC2S = \frac{N(A, B)}{N(A, B_i)} \times 100\%$$

Where,
n= number of services in system
$N(A, B_i)$ = number of connections from service A to service B_i

$$DCSS = \frac{Max - \sum_{u \in V} \sum_{v \in V} d(u,v)}{Max - Min}$$

Where,
Max appears when all nodes in the graph do not connect, Min appears when all nodes in the graph connect to others.

$Max = K*V*(V-1)$

$Min = V*(V-1)$

Used On: Web services
Description: The most vital trait of service oriented software is the coupling. Coupling is represented by the relationship between the services, where this relationship shows the dependency between services. In this study, a set of dynamic coupling metrics has built up for service oriented software which is based on the interaction between the services in a system at runtime. CBS is developed directly from CBO metric. If for a service, the value of CBS metric is higher, the relationship with other services would be more firm. Due to which the maintainability will be also low.
In IMS, the fan.in and fan.out metrics are used for estimating the software's maintainability. If the value will be low then the level of dependency of service will be low and others reckon on it will be higher.
DC2S metric is derived from CBS metric. It pinpoints the level of coupling between two services in the runtime.
DCSS exhibits the coupling between the services in a system. If the coupling is flowing then the system will be simple and inexpensive in maintenance with altering.
Validation: Experimentally (real time system)
Advantages and Disadvantages: The proposed metrics need not use the system's specification as compared to earlier proposed metrics.Even, the suggested metrics also solves the problem by helping users to choose any services at any positions whereas other metrics limits the measuring services.
But the communication between the services in a node is irrelevant, which does not influence the other services in other nodes.
Future Scope: It will continue its work by building a set of performance metrics. Even, it will also build a large storage which will contain the available information about the services.

- **Bela Ujhazi, Rudolf Ference, Denys Poshyvanyk and Tibor Gyimothy**

Year: 2010
Objective: The main objective of this paper is to embody the two novel conceptual metrics for quantifying cohesion and coupling in software systems i.e. CCBO and CLCOM5. And also study empirically the parameters which can affect the performance of CCBO and CLCOM5 metrics.
Proposed Metric Used:

$CLCOM5(c,x) = NoCC(G)$

Where,
NoCC=Number of connected components in the graph

$G_c = (M(c), E)$, $c \in C$, $E \subseteq M(c)*M(c)$ and $(m_k, m_j) \in E$ if

$CSM^P(m_k, m_j, t) = 1$

For class c,

$$CCBO(c,t) = \sum_{c_k \in C, c \neq c_k} CSC^{p}(c, c_k, t)$$

Used On: Open source software systems
Description: The proposed metrics also reckon on the alike primary mechanism to extort and scrutinize the conceptual information from the identifiers and comments in the source code as prior the conceptual metrics. These conceptual metrics lean on the Latent Semantic Indexing (LSI). Earlier, it was used to support other source code scrutiny efforts such as identification of abstract data types, clone detection, concept location, traceability recovery links among software artifacts, software clustering, quality estimation and software metages. CLCOM5 metric uses CSMP for the computing conceptual similarities among the classes, on the other hand, while in terms of counting mechanism it relies on earlier defined structural metrics known as LCOM5 (graph based cohesion metric). Whereas, CCBO metric depends on CoCC metric.
Validation: Empirically validated.
Advantages and Disadvantages: The proposed conceptual metrics can be computed in a much simpler way as compared with other structural metrics. The CCBO and CLCOM5 metrics not only be used to build operational models for predicting fault-proneness of classes, even it can also be used efficiently in conjunction with other structural metrics to improve the overall accuracy of bug prediction models.
The proposed measures, CCBO and CLCOM 5 do not explain about the polymorphism and inheritance. They only consider methods of a class which is employed or overloaded in the class. In this study, they have considered one large software system for envisioning fault proneness so to favor for generalization of results, again it needs a large-scale evaluation.
Future Scope: The proposed metrics can be extended further and enhanced by taking into some factors like inheritance. Even the quality of primary textual information can be improved by employing the advanced source code preprocessing techniques for splitting and expanding compound identifiers and comments in the software.

- **Varun Gupta, Jitender Kumar Chhabra**

Year: 2011
Objective: To propose a new measure for gauging of package cohesion based on formal definitions and properties of packages. And, aim to use the new proposed metric as a good predictor of reusability as compared to existing cohesion measures.
Proposed metric used:

$$PCoh(p^i) = \begin{cases} 0, & \text{if}(n = 0) \\ \frac{\sum_{x=1}^{n} \sum_{y=1 \wedge y \neq x}^{n-1} r(ex^{i+1}, ey^{i+1})}{n(n-1)} & \text{if}(n > 1) \\ 1 & \text{if}(n = 1) \end{cases}$$

Where,
n = number of elements at hierarchical level $i+1$ in a package p^i
$p^i = < E^{i+1}, R^{i+1} >$ defined at hierarchical level i
Used on: Open source software systems
Description: The proposed package level cohesion
metric is based on the relations among pairs of package elements. These elements can be classes, interfaces or sub packages even the hierarchical structure of a package is also considered. By using the four properties stated by Briand et. al.- null value and maximum value, monotonicity, non-negative and normalization and merging of unconnected packages for evaluating the proposed measure theoretically. Also, 25 packages are taken from 6 open source projects for evaluation of the proposed metric experimentally. The statistical analysis is used to carry out the correlation of proposed package cohesion metric with package reusability with the help of Karl's Pearson coefficient correlation.
Validation: Theoretically and Experimentally.
Advantages and Disadvantages: This cohesion metric is proved to be the better predictor of code reusability than already existing cohesion metric.
There are threats to the validity of this study with other empirical experiments. It might be that reuse effort may not represent the quality of the package. Somehow, it could not able to establish the cause–effect relationship between package cohesion and package reusability.
Future Scope: The other measures of quality like maintenance effort can be used for validation for reducing above stated threats in future. Need to advance empirical research of the proposed package level metric in turn to establish its relations with other external software quality.

- **Sandip Mal, Kumar Rajnish, and Sanjeev Kumar**

Year: 2013
Objective: To propose a new package cohesion metric (CohP) which is based on basic definitions, properties and relations of elements of a package. Also, aim to extend the packages i.e. extendibility.
Proposed Metric Used:

$$CohP(I) = \frac{(\text{Number of relations between elements of a package})}{n(n-1)}$$

Where,
n = number of elements of package I
CohP value lies between 0(low cohesive) and 1(high cohesive)
Used on: Open source software systems
Description: It is stated that the measure of CohP is based on packages and their structures. The theoretical validation is done by analyzing the mathematical properties with proposed cohesion measures. Briand et al. stated five properties which are used here and provides useful guidelines and helps in the validation of coupling measures in a very precise manner. The empirical validation is done by using two open source software system. A positive Spearman correlation(0.072) shows in between the average effort and a number of the classes added and recorded positive Spearman correlation(0.067) given by the number of classes and CohP.
Validation: Both theoretically and Empirical Validation.
Advantages and Disadvantages: It shows CohP as the reliable indicator for external quality attributes like Extendibility.
It still lacks advance empirical research of the package level metric.
Future Scope: Can be used to analyze the proposed metric with other performance indicators such as design, effort, system performance, and maintenance.

- **Girish K. K**

Year: 2014
Objective: To propose a measure i.e. Conceptual Cohesion of Classes (C3), for class cohesion which details the conceptual aspects of class cohesion. And, quantifies how strongly the methods of class relate to each other conceptually.
Proposed Metric Used:
For a class $c \in C$,

$$C3(c) = \begin{cases} ACSM(c), & \text{if } ACSM(c) > 0 \\ 0, & \text{Else.} \end{cases}$$

Where,

$$ACSM = 1/N * \sum_{i=1}^{N} CSM(m_i, m_j)$$

Used On: Open source software
Description: The proposed measure C3 is based on the study of textual information in the source code which is expressed in comments and identifiers. The source code is somehow closer to natural language but still, it is different from it so classical processing methods are impractical or unfeasible. Thus, Latent Semantic Indexing (LSI) to extract, represent and analyze the textual information from the source code. The proposed measure of cohesion can be explained as the measure of the textual coherence of a class contained by the entire system.
Validation: Hypothetically (based on experiments and findings)
Advantages and Disadvantages: C3 can be used to solve the aspect of object oriented concepts, objects interrelated real world scenarios which most of the structural metrics will not be able to incorporate. C3 gives prominence on the object and this object itself is the 'concept' of conceptual cohesion.
It lacks empirical and experimental validation which may discover some threats.
Future scope: It can work on the indication about the readability and comprehensibility of the source code.

- **Nikita Singh and Aprna Tripathi**

Year: 2015
Objective: To propose a model for measuring understandability using coupling and cohesion.
Proposed Metric Used:
(Package level Coupling)

$$PLC\ for\ P_i = \frac{\sum_{j=1}^{N} \text{weight of connections from } P_j \text{ to } P_i}{N-1}$$

Where, $P_j \in P - P_i$
Let any class C_i belonging to package P,
Mathematically:

$$\text{Package-level Cohesion(PLCoh)} = \frac{\sum_{i=1}^{N} C_i Coh}{N}$$

Where,

$$\text{Cohesion of class } C_i (C_i Coh) = \frac{\text{Number of relationships (directly or transitively related with } C_i)}{(N-1)}$$

N= total number of classes in a package P

Used On: Open source software systems

Description: It becomes very difficult to understand the system when it is reused or maintained by the other programmers which make maintainability very complex. So, understandability has a major contribution to handle maintainability. In this study, a new model is proposed for package-level understandability that relies on coupling, cohesion, and DIT. For evaluation, six Java based open source systems are used. If DIT is lesser, the requisite understandability will be lesser.

Validation: Statistically

Advantages and Disadvantages: The new proposed measure using coupling and cohesion helps in better understandability of the system and improves maintainability.

The study lacks identical experimental trial which can help for better utility.

Future Scope: Furthermore research work can be done in this area of the field for improving maintainability.

- **Calvins Otieno, George Okeyo, and Stephen Kimani**

Year: 2016

Objective: The objective of this study was to develop a unified model for measuring coupling.

Proposed Metric Used:

$$CLC_{(i,j)} = \sum_{i=1}^{i=n} \left(\frac{M_h M_i M_a M_p}{n * c} \right)$$

Where,

C_i : is an instance of a class

C_j: is the set of classes invoking the methods or inheriting methods of other classes

M_i : Number of methods invoked

M_h: Number of methods inherited

M_a: Number of methods referencing attributes of another class

M_p: Number of methods having parameter of the type of another class.

$$OLC_x(O_i O_j) = \sum_{z=1}^{z=n} \left(\frac{M}{Z} \right)$$

Where,

o_i: an instance of a class (an object)

O: is the set of objects collaborating during the execution of a specific scenario

|Z|: number of elements in set Z.

$$DLC = (D_{ca} + D_{kc} + P_{ac}) / TC$$

Where,

D_{ca} : Dynamic Afferent Coupling

D_{kc}: Dynamic Key Coupling

P_{ac}: Percentage Active Class

$$SLC = \sum_{c=i}^{c=n} \left(\frac{I_m}{T_m} \right) TTC$$

Where,

I_m: The set of all interacting modules in the class under study

T_m : The set of total modules existing within the class under study

TTC: The total number of classes to be studied.

Used On: Software Systems

Description: The proposed model come across through class coupling level, static coupling level, dynamic coupling level and object coupling level due to which it would be more efficient in measurement. The data collected were evaluated in Poisson model and generates beta values. The count data produced in the research was evaluated by the Poisson model.

Validation: Empirically

Advantages and Disadvantages: It provides an algorithm which can access the whole coupling. It helps the reviewers and assessors in knowing whether the software is highly coupled or loosely coupled.

Still, it lacks somehow standardization for stating coupling.
Future Scope: It can work on conclusive the effect of coupling in whole software maintenance and debugging.

IV. DISCUSSION

We have created a comparison table for metrics of coupling and cohesion based on our evaluation criteria. We have studied some appropriate existing metric for it in object oriented design. A tabular discussion is shown in below Table 2.

Table2: The existing metrics against structured attributes

AUTHOR/ STUDY	YEAR	METRIC OBJECTIVE	DOMAIN OF MEASURE	PROPOSED METRIC	APPLICABLE/ USED ON	UML	VALIDATION	ADVANTAGES/DISADVANTAGES	FUTURE SCOPE
Chidamber et. al.	1994	The main objective of this paper was to have expansion and empirical validation of a set of theoretically grounded metrics of OO design.	Medium Grain	CBO RFC LCOM	Large software systems	No	Theoretically (analytical) and Empirically	It was the first empirically validated plan for formal metrics of OOD. Somewhere C-K metrics had theoretic anomalies.	Evaluate the extent to which these metrics correlate with managerial performance indicators. To chase commercial applications from outset to development and congregate metrics at varied intermediate stages of the project.
Fernando Brito e Abreu et. al.	1994	To put forward a set of metrics for accessing the use of the main generalization of the OO paradigm such as inheritance, information hiding and gives focus on reuse.	Medium Grain	COF	Real Time Projects	No	Heuristically	It helped in setting OO design standards at the organization level. Also, helped OO specialists to direct their development process. The drawback was the values used were inappropriate.	To combine MOOD metrics except reuse factor to achieve generic OO software system complexity metric. To study the statistical independence of each MOOD metric towards each other.

James M. Bieman and Byung-Kyoo Kang	1995	To assess the relationship between class cohesion and private reuse in the system. And build up the cohesion measures and relate them to a large C++ system.	Medium Grain	TCC LCC	Large C++ systems	No	Experimentally (validated on real time system-Interview System, Stanford University)	The proposed measures can be used to locate classes which have been designed inappropriately. The results show that classes which have been heavily reused by using inheritance exhibit lower cohesion.	The results will be confirmed by the study of additional software systems.
Pham Thi Quynh, Huynh Quyet Thang	2009	To propose a suite of dynamic coupling metrics of service-oriented software.	SOA	CBS IMS D2SS DCSS	Web Services	No	Experimentally (real time system)	Proposed metric need not to use system's specification. Solves the problem by helping users to choose any services at any positions. But, communication between the services in the node is irrelevant.	It will continue its work by building a set of performance metrics. Even, will build a large storage which will contain the available information about the services.
Bela Ujhazi et. al.	2010	To embody the two novel conceptual metrics for quantifying cohesion and coupling i.e. CCBO and CLCOM5. And also study empirically the parameters which can affect the performance of the metrics.	Medium Grain	CCBO CLCOM 5	Open source software systems	No	Empirically	Can be computed in a much simpler way as compared with other structural metrics. The proposed measures, CCBO and CLCOM 5 do not explain about the polymorphism and inheritance.	It can be extended further and enhanced by taking into some factors like inheritance. Even the quality of primary textual information can be improved.

Varun Gupta et. al.	2011	To propose a new measure for gauging of package cohesion based. And, aimed to use it as a good predictor of reusability as compared to existing cohesion measures.	Coarse Grain	PCoh	Open source software systems	No	Theoretically and Experimentally	It is proved to be the better predictor of code reusability than already existing cohesion metric. There are threats to the validity of this study with other empirical experiments.	The other measures of quality like maintenance effort can be used for validation for reducing threats. Need to the advance empirical research of the proposed package level metric to establish its relations with other external software quality.
Sandip Mal et.al.	2013	To proposed a new package cohesion metric (CohP) & also aimed to extend the packages i.e. extendibility.	Coarse Grain	CohP	Open source software systems	No	Theoretically and Empirically	It shows CohP as the reliable indicator for external quality attributes like Extendibility. It still lacks advance empirical research of the package level metric.	Can be used to analyze the proposed metric with other performance indicators such as design, effort, system performance, and maintenance.
Girish K.K	2014	To propose a measure i.e. Conceptual Cohesion of Classes (C3), for class cohesion. And, quantifies how strongly the methods of class relate to each other	Medium Grain	C3	Open source Software	No	Hypothetically (based on experiments and findings)	C3 can be used to solve the aspect of object oriented concepts. It lacks empirical and experimental validation which may discover some threats.	It can work on the indication about the readability and comprehensibility of the source code.

		conceptuall y.							
Nikita Singh and Aprna Tripath i	201 5	To propose a model for measuring understand ability using coupling and cohesion.	Coarse Grain	PLC PLCoh	Open source software systems	No	Statisticall y	It helps in better understandability of the system and improves maintainability. It lacks identical experimental trial which can help for better utility.	Further, more research work can be done in this area of the field for improving maintainabi lity.
Calvins Otieno, George Okeyo, and Stephe n Kimani	201 6	The objective of this study was to develop a unified model for measuring coupling.	Mediu m Grain	CLC OLC DLC SLC	Software systems	No	Empiricall y	It provides an algorithm which can access the whole coupling and help reviewers and assessors in knowing whether the software is highly coupled or loosely coupled. Still, it lacks somehow standardization for stating coupling.	It can work on conclusive the effect of coupling in whole software maintenanc e and debug.

V. CONCLUSION

The main involvement of this work is to provide an evaluation of some related works to the metrics for cohesion and coupling in object-oriented systems. We have proposed an attribute-based structure to allow gauging cohesion and coupling at all three levels- class level, method level and package level. This scrutiny works as a guide for researchers who are engrossed in developing cohesion and coupling metrics. In future, following up this work we will design some new metrics for coupling and cohesion which will help in overcoming some stated drawbacks in our study.

VI. REFERENCES

[1] S. Chidamber and C. Kemerer "A Metrics Suite For Object Oriented Design", IEEE Transaction on Software Engineering, Vol.20, No.6, pp.476-492, June 1994.
[2] F.B. Abreu and R. Carapuça "Object-Oriented Software Engineering: Measuring and Controlling the Development Process", In the Proceedings of 4th International Conference on Software Quality, McLean, VA, USA, October 1994.
[3] J.M. Bieman and B.K. Kang "Cohesion and Reuse in an Object-Oriented System", In the Proceedings ACM Symposium on Software Reusability (SSR'95), pp. 259-262, April 1995.
[4] L.C. Briand, J.W. Daly, J. Wust "A Unified Framework for Cohesion Measurement in Object-Oriented Systems", IEEE Technical Report, ISERN-97-05, 1997.
[5]Rachel Harrison, Steve J. Counsell and Reben V. Nithi "An Evaluation of the MOOD Set of Object Oriented Software Metrics", IEEE Transactions on Software Engineering, Vol.24, Issue 6, pp.491-496, June 1998.
[6] P. Thi Quynh, H.Q. Thang "Dynamic Coupling Metrics for Service – Oriented Software", International Journal of Electrical and Electronics Engineering, 3, pp.282-287, June 2009.
[7] Bela Ujhazi, Rudolf Ference, Denys Poshyvanyk and Tibor Gyimothy "New Conceptual Coupling and Cohesion Metrics for Object-Oriented Systems", IEEE 13th International Working Conference on Source Code Analysis and Manipulation (SCAM), ISBN: 978-0-7695-4178-5, pp. 33-42, September 2010.

[8] I. Kaur, N.Kaur, A. Ummat, J. Kaur, N. Kaur "Research Paper on Object Oriented Software Engineering", International Journal of Computer Science And Technology (IJCST), ISSN(Online): 0976-8491, Vol. 7, Issue 4, pp.36-38, October/December 2016.
[9] Amjan Shaik, Dr. C. R. K. Reddy, Dr. A. Damodaram "Object Oriented Software Metrics and Quality Assessment: Current State of the Art", International Journal of Computer Applications (0975 – 8887), Vol. 37, No.11, pp.6-15, January 2012.
[10] V. Gupta, J.K. Chhabra "Package Level Cohesion Measurement in Object-Oriented Software", Journal of Brazilian Computer Society, 18, pp.251-266, December 2011.
[11] Singh N., Tripathi A."A Design Phase Understandability Metric Based on Coupling and Cohesion for Object-Oriented Systems", Advances in Intelligent Systems and Computing, Springer, Vol. 308, 2015.
[12] Sandip Mal, Rajnish Kumar, Sanjeev Kumar "Package Level Cohesion Metric for Object-Oriented Design", International Journal of Engineering and Technology (IJET), ISSN: 0957-4024, Vol.5, No.3, June/July 2013.
[13] Girish K. K."Conceptual Cohesion of Classes(C3) Metrics", International Journal of Science and Research (IJSR), ISSN (Online): 2319-7064, Volume 3 Issue 4, April 2014.
[14] C. Otieno, G. Okeyo and St. Kimani "A Unified Coupling Model for Coupling Measurement in Object Oriented Software Systems", The International Journal Of Engineering And Science (IJES), Vol. 5, Issue 2, pp.37- 44, ISSN (e): 2319 – 1813 ISSN (p): 2319 – 1805, 2016.

APPENDIX

OO	Object Oriented
LCOM	Lack of Cohesion in Methods
RFC	Response for Class
CBO	Coupling between Objects
COF	Coupling Factor
MOOD	Metric for Object-Oriented Design
TCC	Tight Class Cohesion
LCC	Loose Class Cohesion
SOA	Service Oriented Architecture
CBS	Coupling between Service
IMS	Instability Metric for Service
DC2S	Degree of Coupling between 2 services
DCSS	Degree of Coupling within a given set of services
CCBO	Conceptual Coupling between Object Classes
CLCOM5	Conceptual Lack of Cohesion on Methods
PCoh	Package Cohesion
C3	Conceptual Cohesion of Classes
ACSM	Average Conceptual Similarity of Methods
DIT	Depth of Inheritance Tree
OLC	Object Level Coupling
DLC	Dynamic Level Coupling
CLC	Class Level Coupling
SLC	Static Level Coupling