International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-2, December, 2019

Speculation of Software Reusability Estimation using CK (Chidamber and Kemerer) Metrics

Ridhi Jindal, S. K. Mittal

Abstract*: *In today's software community the most interesting topic is software reusability because of its immense benefits that comprise of decreased product schedule, cost and increase in product quality. Most of the time, software is not built from scratch since it is costly and time-consuming process. Therefore, existing software documents (source code, documents, design, etc.) are used to develop the new application according to user requirements. But still the software reusability is not being followed as a standard approach in the process of software development. Till now initiating the software reuse process there is a need to analyze and properly understand the user requirements in spite of considerable upfront investments for software reusability. We have studied various aspects of software reusability along with software metrics and are being presented in this article. Efficient software designs can be enabled by assessing the software reusability extent. The aging resilient software design could be of paramount significance to enable faultiness software system. The estimation of software reusability plays an important part in software's cost reduction and quality improvement, in an object-oriented programming. In this paper the idea about the designing the CK metrics suite along with metrics' evaluation is presented that can help for object-oriented based systems in reflecting the accurate results.

Keywords*: *Software reusability, web of services, Software development, CK metrics

I. INTRODUCTION

Software reusability is defined as the process involved in software development when a software system is updated or implemented using already existing software component. By using a suitable software reuse process the product reliability, quality, and productivity are increased whereas implementation time and cost is decreased. As to initiate a software reuse process there is a requirement of an initial investment which in a few reuses pay for itself. A knowledge base is produced by the development of repository and reuse process which helps for future works in improving the quality, reducing the development work and finally, the repository knowledge-based projects risks are also reduced. During the software development implementation, the significant characteristics are used for satisfying the assured attributes of software quality, for supporting the necessary quality standards. The understandability and maintainability are considered as the main components of software metrics.

Revised Manuscript Received on December 30, 2019.
* Correspondence Author
Ridhi Jindal*, Department of Computer Science & Engineering, Rayat Bahra University, Mohali, Punjab, India. Email: ridhijindal136@gmail.com
Dr. S. K. Mittal, Department of Computer Science & Engineering, Rayat Bahra University, Mohali, Punjab, India. Email: skmskm1@rediffmail.com

Re-coding of the work is avoided by a significant technique that is known as software reusability which can be used for improving the software development system's quality [1]. The principles of object-oriented give the idea about the estimation tools that could be applied and recognized for software development. The quality as well as performance along with the system maintenance of SDLC (Software Development Life Cycle) is increased by the software component reusability [2-4]. The amount of effort is reduced which is required in case to develop a software from the scratch, therefore, the testing time required is to be less for new software. With the help of reuse strategy industrial observer suggest that the development cost could be reduced up to 20% of actual development costs [5]. Software reusability is a process of designing the new software from the existing modules [6]. It has various advantages in [7] manners like are to reduce Cost of development, Highly Reliable, Quality of service, Time consumption in design, and Maintenance cost.

II. SOFTWARE REUSABILITY

A. Software reuse approaches

There are a number of approaches in reusing software reported by the researchers in the literature. We can broadly divide these reuse approaches into three broad categories:
(i)Component-based software reuse,
(ii) Domain engineering and software product lines and
(iii) Architecture based software reuse.

Component-based software reuse: In component-based reuse, a repository of small independent software components is built and a searching mechanism to match the requirement with stored components is also developed. Finding the best match is the first and most important step in reuse as suggested by Manhas et al. [3]. For example, in mechanical engineering, some parts of an existing machine can be used to fabricate a new machine. Like it, some components of one software systems may be used to build another software system. A software component is defined as the prewritten software element having a well-defined interface as well as proper functionality which classifies its interaction and behavioral mechanism. It was identified by the McClure [5] that most of the software component properties should be reusable. One of the greatest issues of such approach is repository management of the reusable components and developing an efficient retrieval mechanism.

Domain engineering and software product lines: In a software system set the variability and commonalities are captured by the domain engineering so that they can be used for building the reusable assets. Mainly the various organizations are functionally active in specific domains.

For a specific domain area, a few systems are connected as a family according to the customer requirements. Domain engineering mainly identifies the most basic characteristics of the already existing systems and according to these characteristics a new system is developed in that particular domain [2]. This results in greater productivity and efficiency. This particular approach of domain engineering basically works in two stages: domain analysis and domain implementation. In the first stage, the particular system is examined to discover the variability and commonalities in a particular domain, and process is known as domain analysis. In the second stage, the particular information of variability and commonalities gained from domain analysis is implemented for developing the reusable assets and further these reusable assets are utilized for developing the new systems in that domain. Various domain engineering approaches are DAREm, FAST, FORM, KobrA, PLUS, etc.

Architecture based software reuse: Effective reuse depends not only on finding and reusing components but also on the ways those components are combined [16]. The architecture of a software system is composed of its software components, their external properties, and their relationships with one another. Shaw [51] classified software architecture into common architectural styles where every style has four major elements: components, connectors, a control structure, and a system model. Connectors mediate interactions among components. Software architecture may be based on services. This leads to a new approach known as Service-Oriented Architecture (SOA). SOA brought new chances to improve the development of reusable components

B. Software reuse levels

In the software industry, the reusability of software is still an emerging field. There are many forms of reusability that comprise white box reuse to black box reuse, as well as ad hoc reuse to systematic reuse.

Enterprise-level reuse: In this particular model, reusable resources are considered as corporate resources. Architecture coordination is provided by the reuse support organization and also libraries asset and its staff are managed and to make sure that the libraries consistently satisfy projects' requirements [11].

Inter-Project reuse: In this particular model, central support is provided that encourages the reuse among projects. There's additionally a matching role that tries to deliver tasks in concert to simplify reuse process. This demands nonetheless more energy and discipline, though the advantages aren't restricted to individual jobs.

Intra-Project reuse: Such a model performs the reuse within tasks. For the business, reuse is more dynamically urged, and thus there might be main staffs helping to educate, motivate, as well as also help tasks attempting to reuse. This usually takes somewhat much more energy, but with that hard work, several tasks can become destinations of reuse results.

Ad-hoc reuse: Such totally decentralized, ad-hoc design, promotes reuse as an extract objective, but there's absolutely no strategy, without control, so absolutely no assistance from the business as a whole. This's really convenient to attain, though it seldom would make a noticeable impact.

C. Types of Reuse

Vertical reuse: Vertical reuse, drastically unexploited by the program group by huge, but likely extremely helpful, has much-reaching ramifications for future and current application development efforts. The primary concept is the reusing functional areas of the system, or maybe domains which may be utilized by a family unit of devices with comparable efficiency. The application, as well as research of this idea, has spawned additional engineering goals, such as domain name and application engineering which are totally opposite to each other. Domain engineering concentrates on the formation as well as upkeep of reuse repositories of purposeful places, while software engineering uses those repositories to carry out items that are new [13].

Horizontal reuse: Horizontal reuse describes software program pieces employed across a number of uses. In terms of code property, that contains the usually planned library components, like a string manipulation regime, related show type, or maybe GUI (Graphical User Interface) events. It may also relate to the usage of COTS (Commercial Off-The-Shelf) or maybe a third-party program within a bigger structure, like an e-mail bundle or maybe a word processing program. A number of repositories and program libraries that contains the code type, as well as proof, are available these days at numerous web locations [12].

D. Layers of Reusability

- **Implementation layer:** The reusability, programs and APIs are used in the new model. At this level code and generators are to be used.
- **Design Layer:** Object design is to be reused in the new system by containing layer transformation in the old to produce the latest designs. Design layer supports the component-based software development model.
- **Architect Layer:** In this layer, we reuse the software architect, classes and interface.
- **Whole system:** In this layer various application is to be used to support the reusability these are: Enterprise resource planning, Software product line, Commercial off the shelf.

E. Factors affecting Software reusability

Software quality attribute cannot be measured directly. They depend on the following factors:

(a) Modularity: This is a term in which the components are divided into smaller parts. These parts are able to do work independently. The modularity range lies between 0 and 1. When the value of modularity is high then the reusability is better.

(b) Maintainability: It should be high for the better quality of service of the software.

(c) Flexibility: It is the degree when the changes can be made in the software.

(d) Adaptability: The ability of the software component to adopt the new one. For better reusability, it should be high

(e) Interface complexity: By using interface the interaction between the software and the application is possible. There is a difficulty for reuse when the complexity interface is high.

III. LITERATURE REVIEW

Nor et al, 2004, [12] presented the reusability approaches for load-flow analysis in a computer program. In reusability algorithms and codes were obtained by matrix partitioning method.

The software was developed by using a component-based approach an object-oriented approach. This approach makes possible and easy to add, change and update the algorithm without affecting the other code. Rotaru, et al., 200, [13] Cluster Computing Conference on Computer Systems and Applications presented the reusability metrics for the software components. The main goal of this work is to study the compose-ability and adaptability of the software component in a qualitative and quantitative manner. In this study, a mathematical model and a metrics for a software component are presented. The study helps to define the metrics on a functionality basis. Gill, Nasib, et al, 2006, [14] presented a survey on the component-based software development process and discussed the issues and their solution. Testing of the third-party software component is a very complex task. The software reusability and characterization provide an effective architecture, retrieval, usage, and better cataloging. This article helps the developer in software reuse process which is most important in component-based software development. Qureshi et al. 2008, [15] presented a component-based software development process model. It defines the importance and role of the repository in component-based development. The result of the study concluded that CBD is more cost-effective, time-saving and provides smooth development in the software. Saglietti, et al., 2009, [16]: worked on the integration and reliability testing on the CBS system. This sharpness enhancement process is also proposed in this reliability system. It reduces the effort of verification and validation techniques. Complexity is slightly reduced but not affecting more to maximize the saving. Mohamed et al., 2010, [17]: presented an approach of fault tolerant for reliability assessment. It effectively shows the impact of failure on reliability. Hsu, et al., 2011, [18]: In this paper, for a modular software system an adaptive framework of incorporating path into reliability prediction is given by the author. To compute the reliability of the path authors introduces three estimated methods which are based on common program structure, namely, sequence, branch, and loop structure. These computed paths are implemented on predicted software reliability. There are several experiments are executed on the basis of two real systems. Accuracy and correlation are determined by using simulation and sensitivity analysis. Zhang et al., 2012, [19]: presented a method for software reliability estimation by using a Markov model using importance sampling. The importance of sampling is a method to measure the change of measure and speeding up the simulation process. It avoids the state space explosion and gives pre-information on failure probabilities. Singh, et al., 2013, [20]: presented a model for reliability estimation on the component-based system. The reliability is estimated by using path probability and impact factor of the component. It estimates the reliability after the integration of the components and checks the contribution of each component when they are activated. CBS (Component Based Software) reliability is increased by using impact factor analysis. On reusability. Tyagi, et al., 2014, [21]: There are numbers of approaches introduced to predict CBSE (Component Based Software Engineering) reliability. In this paper, the author introduces a model known as the adaptive neuro-fuzzy inference system (ANFIS) to predict the CBSS (Component Based Software System) reliability on the basis of neural networks and fuzzy logic. There is a comparison of the performance with that of a plan FIS (Fuzzy Inference System) for different datasets. Brosig, et al., 2015, [22]: In this paper, there is an in-depth comparison and quantitative evaluation of representative mode transformations. Here there is imparting of the semantic gaps between typical source model abstraction and the various analysis techniques. In this paper, to evaluate the accuracy and efficiency of every individual transformation by using four case studies representing systems of various size and complexity. The result and idea from evaluation help to choose proper transformation for a given context especially for software architects and performances engineers. Also, results in the effective improvement in the usability of model transformation to estimate the performance. Tyagi, Kirti, et al, 2016, [23]: proposed a reliability estimation model which is based on the fuzzy TOPSIS (Technique for Order Performance by Similarity to Ideal Solution). The fuzzy approach is used to order the preferences of the components that are a similar and ideal solution. The linguistic variables are used for weight criteria and ratings of the alternatives in terms of triangular fuzzy numbers. The proposed approach effectively ranks the components on the basis of their reliability. Singh, Neha, et al., 2017, [24]: presented a similarity-based approach for reliability estimation of SOA (Service-Oriented Architecture) system by using a fuzzy approach. The proposed approach is based on the ranking of each service and selects the best for the estimation. The factor that affects reliability is also presented. Zapata et al, proposed a software requirement catalog for mobile application related to health. These applications were used to develop the new enhanced application. This work based on the SIREN (SImple REuse of software requiremeNts) methodology to create the reusable catalog. The audit method is used to check the verification of the catalogs.

IV. RESEARCH METHODOLOGY

This particular section primarily discusses the object-oriented software metrics and their significance towards the software reusability assessment. The used matric are the primary aspects for the research carried out.

A. Object-oriented software metrics

Software metrics are one of the most important tools used in software engineering to assess and optimize the quality of the software. Software metrics are applied throughout SDLC to assist, to estimate, quality control, productivity assessment and for the project control. In this paper different object-oriented CK metrics suite has been taken into consideration to predict software component reusability defined as a function of URL (Uniform Resource Locator) extracted features. Mathematically reusability is defined as

Reusability = f (URL based extracted features)

B. Metrics set

Considering specific OOP (Object Oriented Programming) based software metrics and their significance to characterize the software quality and software component reusability. In this paper, six CK metrics have been extracted from the classes. The object-oriented metrics have been used and are presented in table I.

C. Effectiveness of metrics for reusability assessment

Determining the reusability data, we intend to formulate the relationship between object-oriented CK metrics and the software reusability. The software reusability is considered as a dependent variable, while the individual metrics are considered as an independent variable. Mathematically object-oriented six CK metrics can be defined as:

Reusability = f (WMC, DIT, NOC, CBO, RFC, LCOM)

Table- I: Object-oriented CK metrics for reusability estimation

S. no.	Object-oriented CK metrics	Definition
1	WMC (Weighted Methods for Class)	In a software project, it denotes the sum of each and every class method's complexity.
2	DIT (Depth of Inheritance Tree)	It denotes the greatest length to the node from the root through a tree.
3	NOC (Number of Children)	In the associated class hierarchy, it denotes the complete amount of adjacent subclass that are branches of a class.
4	CBO (Coupling Between Objects)	It indicates how suitably the classes are interconnected with one another.
5	RFC (Response For Class)	It indicates a method that is implemented in a response to a particular message that is attained by that class's object.
6	LCOM (Lack of Cohesion of Methods)	It indicates the dissimilarity or divergence among the class methods via instanced variables

V. SOFTWARE REUSE METRICS

Particular object's attributes are quantitatively indicated with the help of metrics whereas the relation between these metrics is specified by a model [23]. According to Frakes [23], reuse metrics as well as models are classified as: (1) Reuse library metrics, (2) Reusability, (3) Failure modes, (4) Amount of reuse, (5) Maturity assessment, and (6) Reuse cost benefits models.

Various object-oriented metrics are to be built, such as, MOOD metrics [28], Li and Henry [27] metrics, CK metrics [26], Abreu proposed metrics [25]. The most popular metrics from all of them is the CK metrics. The CK metrics suit is among the most popular object-oriented design for the system used for complexity measurement that evolves in the software package.

A. Overview of CK metrics

A brief description of the CK metrics suite for the object-oriented design [29, 30] indicates the deepest research in the object-oriented metrics investigation. Object-oriented CK metrics for reusability estimation are explained below in detail. Lack of Cohesion in Methods, coupling between object, WMC, NOC and DIT are used for developing a software component reusability estimation model.

B. Weighted Method per Class (WMC)

In a software project, it denotes the sum of each and every class method's complexity. The method as well as complexity amount implicates a predicator that tells the amount of effort as well as time required to maintain the weight method per class.

When the larger the number of students is in the class, the children experiences the greater potential impact. As all the defined methods of the class are inherited by the children so when the classes are to have a huge method then the classes are more application specific, which then limits the reuse possibility.

C. Depth of Inheritance Tree (DIT)

It denotes the greatest length to the node from the root through a tree. The tree behaviour cannot be determined easily if in a hierarchy, class is presented at a deeper level with greater method number.

As their present large amount of methods and classes in a deeper tree thus it results in a more complex design. Whereas in a hierarchy a deeper class has high potential for inherited methods reusability.

D. Number of Children (NOC)

The instantaneous subclasses number indicates the NOC. One of the main reusable form is inheritance. In case of large number of children, there are possibilities that parent class is abstracted improperly.

The number of children presents the idea about the class's potential influence on the design. Additional testing method of the class is to be required if the class contain a number of children.

E. Coupling between Object Classes (CBO)

It states that how well the classes are connected together. Unnecessary coupling among the classes is detrimental to the modular design which prevents the reuse. If the class is independent then it is easy to reuse it into another application. So, for promoting inter-object, encapsulation and improving the modularity coupling must be least. The greater the sensitivity if the number of couples is to be large and then the maintenance becomes difficult.

F. Response for a Class (RFC)

It indicates a method that is implemented in a response to a particular message that is attained by that class's object. During the testing time the suitable allocation is assisted by the possible worst-case value.

G. Lack of Cohesion in Methods (LCOM)

In the lack of Cohesion, the number of the different methods in the class that a reference is given to the instance variable. Good class subdivision indicates the high cohesion. The cohesiveness of methods in the class desirable because it promotes the encapsulation. Classes should be split into two or more subclasses. Low cohesion increases the complexity. For identifying the objects relation, operations and attributes, and classes and objects on the initial phases of project life cycle a substantial effort is required by the object-oriented methodologies. CK suites covers are utilized for several reasons: CK suite covers all aspects of object-oriented software's these are Reusability, Encapsulation, and Polymorphism.

CK suite was chosen by the Software Assurance Technology Centre at NASA (National Aeronautics and Space Administration) Goddard Space Flight Centre [31, 32] and still used widely.

Much effort was devoted to the empirically validating [33-35] that the original CK metrics and then linking them to object-oriented design quality parameters. Most of the other metrics are formed by the original CK metrics suite. It is very easy to lift the CK metrics from the coded level to the modeling level [36]. CK suite could be linked to the three economic variables. These are Productivity, Rework Effort, and Design.

Table- II: Guidelines "for CK Metrics

METRIC	GOAL	LEVEL	COMPLEXITY (To develop, to test and to maintain)	RE-USABILITY	ENCAPSULATION, MODULARITY
WMC	Low	▼	▼	▲	
DIT	Trade-off	▼	▼	▼	
		▲	▲	▲	
NOC	Trade-off	▼	▼	▼	
		▲	▲	▲	
CBO	Low	▼	▼		▲
RFC	Low	▼	▼		
LCOM	Low	▼	▼		▲

VI. COMPARATIVE ANALYSIS OF THE EXISTING MODELS

Depending on the reuse metrics and various factors few reusability estimation model are compared in this section. The results of the comparisons are presented in, Table III.

VII. SUMMARIZED RESULTS

Based on the literature survey some important observations about the software reusability are as:

- In the object-oriented design's software reusability estimation, one of the significant steps is of design phase, therefore, the various software reusability metrics should be selected.
- Software characteristics must be identifiable.
- For the object-oriented design, minimal set of the software reusability is identified so that efforts used in measuring the object-oriented design's software reusability can also be reduced, which directly impacts the software reusability measurement.
- The software quality is improved with use of software development life cycle as well as if software reusability is estimated at an early stage then time, and cost can be reduced as per the requirements of the customer.

The Evaluation of Chidamber and Kemerer Metrics against weyukar's nine axiom is shown in table IV where the compatibility is represented by Y and N, which states Yes or No.[52] Table V represents the average values for Chidamber & Kemerer metrics [53] based on the analysis of 3 systems and quality is classified. This Quality analysis is based on the 3 languages as specified by NASA which states that higher the CBO and WMC, the lower the quality of the system.

VIII. CONCLUSION

In the software engineering research, evaluation of software techniques as well as software performance, the software metrics plays a significant role. The main aim of this paper is to briefly explain some significant researcher's work that had influential contribution in the field of software concerning the software reusability using software metrics. The software reusability is considered at the formative stage. Significant research opportunities exist in all of the areas of software reusability world. A systematic literature review of software reusability estimation model is carried as well as their result are examined as critical observations in this paper. Related work and the classification schemes serve as the framework for future research to differentiate between the different software reusability estimation to test more models and metrics in practice. The comparative analysis of the existing models and proposed the models discussed in this paper which clearly indicates that the performance analysis in software reusability estimation using CK metrics is better.

Table- III: Comparative analysis of existing models

Year	Authors	Models	Method
2005	Richard W. & Selby [37]	Evaluation Software Reuse Empirically by Mining Software Repositories	Goal Question Metric (GQM)
2006	Parvinder S. & Sandhu [38]	Reusability Evaluation Model	Neuro-Fuzzy Inference System
2007	Parvinder S. & Sandhu [39]	Quantitative Investigation Model	Taguchi Approach
2008	Gui Gui & Paul D. Scott [40]	Evaluation of software component reusability model	Linear Regression and Rank Correlation
2009	Parvinder S. & Sandhu, Harpreet Kaur [41]	Reusability Evaluation System	Neural Network Approaches
2010	Sonia Manhas & Rajeev Vashisht [42]	Reusability Evaluation Model	Neural Network Algorithms
2011	Nasib S. Gill & Sunil Sikka [43]	Inheritance hierarchy Based model	Metrics based approach
2012	Fazal-e-Amin & Ahmad Kamil Mahmood [44]	Reusability Attribute Model	Goal Question Metric (GQM)
2013	Ajay Kumar [45]	Reusability classification Model	SVM classifier
2014	Neha Goyal & Deepali Gupta [46]	Reusability Calculation	CK Metric
2015	M. Huda et al. [5]	Reusability Quantification Model	ANOVA
2016	Amjad Hudaib [50]	Self-Organizing Map (SOM)	CK Metric
2017	Neelamadhab Padhy et al. [49]	Aging Resilient Software Reusability Prediction Model	Web of Service (WoS)
2018	Neelamadhab Padhy et al. [48]	EC-AI-Based Regression Analysis for Reusability Estimation	CK Metric
2019	Neelamadhab Padhy et al. [49]	Model for classification of reuse-proneness or non reuse- proneness classes	ANN, MARS, and EC

Table- IV: The Evaluation of Chidamber and Kemerer Metrics against weyukar's nine axiom

CK Metrices	WEYUKAR'S NINE AXIOMS								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
WMC	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**N**	**Y**	**N**
DIT	**Y**	**Y**	**Y**	**Y**	**N**	**Y**	**N**	**Y**	**N**
NOC	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**N**	**Y**	**N**
RFC	**Y**	**Y**	**Y**	**Y**	**Y**	**N**	**N**	**Y**	**N**
LCOM	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**N**	**Y**	**N**
CBO	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**N**	**Y**	**N**

Table V: Quality analysis for 3 system (NASA)

System Analysed	*JAVA*	*JAVA*	*C++*
Classes	46	1000	1617
Lines	50000	30000	50000
Quality	Low	High	Medium
CBO	2.48	1.25	2.09
LCOM1	447.65	78.34	113.94
RFC	80.39	43.84	28.60
NOC	0.07	0.35	0.39
DIT	0.37	0.97	1.02
WMC	45.7	11.10	23.97

REFERENCES

1. Abdullah, Dr. Reena Srivastava, and M. H. Khan Testability measurement framework: design phase Perspective. International journal of advanced Research in computer and communication Engineering Vol.3, Issue 11 2014.
2. Abdulla'h, Dr. Reena Srivastava, and M. H. Khan Modifiability: A key Factor to testability, International Journal of Advanced Research in information and technology, vol. 26, June 2014.
3. Sonia Manhas, Rajeev Vashist, Parvinder S. Sandhu and Nirvair Meeru, reusability Evaluation Model for Procedure Based Software Systems, International Journal of Computer Electrical Engineering, Vol. 2, No. 6 Dec2010.
4. Abdullah, Dr. M. H. Khan, Testability Measurement Model for Object Oriented design, International Journal of Computer Science and information technology vol. 7, No. 1 February 2015.
5. Huda, M., Arya, Quantifying Reusability of object Oriented design: A Testability Perspective. Journal for software Engineering and applications.
6. Sommerville, Software Engineering 9th Addison–Wesley, New York (2011).
7. Goel B.M, Bhatia, and P.K: Analysis of reusability of object-oriented systems using object-oriented metrics. (2013).
8. P.K. Bhatia, Research on software reuse methods based on the object-oriented components, Computer Science and Network Technology vol. 5, 2012.
9. Peter Freeman, 1983. Reusable software Engineering concept and research directions.
10. Dromey RG. Concerning the Chimera (Software quality) IEEE software
11. Boehm BW, Brow JR, Lipow M, Mcleod G, Merritt, Characteristics of software reusability 1978.
12. Nor, Khalid M., Hazlie Mokhlis, and Taufiq Abdul Gani. Reusability techniques in load-flow analysis computer program. IEEE Transactions on Power Systems 19.4 (2004): 1754-1762.
13. Rotaru, O.P., Dobre, M.: Reusability metrics for software components. In: Proceedings of the 3rd ACS/IEEE International 123 Cluster Computing Conference on Computer Systems and Applications, pp. 24–29 (2005).
14. Gill, Nasib S. Importance of software component characterization for better software reusability. ACM SIGSOFT Software Engineering Notes 31.1 (2006): 1-3.
15. Qureshi, M. Rizwan Jameel, and S. A. Hussain. A reusable software component-based development process model. Advances in engineering software 39.2 (2008): 88-94.
16. Saglietti, Francesca, Florin Pinte, and Sven Söhnlein. Integration and reliability testing for component-based software systems. 35th Euromicro Conference on Software Engineering and Advanced Applications, 2009. SEAA'09. IEEE, 2009.
17. Mohamed, Atef, and Mohammad Zulkernine. Failure type-aware reliability assessment with component failure dependency. 2010 Fourth International Conference on Secure Software Integration and Reliability Improvement (SSIRI). IEEE, 2010.
18. Hsu, Chao-Jung, and Chin-Yu Huang. An adaptive reliability analysis using path testing for complex component-based software systems. IEEE Transactions on Reliability 60.1 (2011): 158-170.
19. Zhang, Deping, Shuai Wang, and Wujie Zhou. Software reliability estimation method based on markov usage models using importance sampling. 2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI). IEEE, 2012.
20. Singh, Aditya Pratap, and Pradeep Tomar. A new model for reliability estimation of component-based software, 2013 IEEE 3rd International on Advance Computing Conference (IACC), IEEE, 2013.
21. Tyagi, K., & Sharma, A. (2014). An adaptive neuro-fuzzy model for estimating the reliability of component-based software systems. Applied Computing and informatics, 10(1), 38-51.
22. Brosig, F., Meier, P., Becker, S., Koziolek, A., Koziolek, H., & Kounev, S. (2015). Quantitative evaluation of model-driven performance analysis and simulation of component-based architectures. IEEE Transactions on Software Engineering, 41(2), 157-175.
23. Tyagi, Kirti, and Arun Sharma. Ranking of components for reliability estimation of CBSS using fuzzy TOPSIS. International Journal of System Assurance Engineering and Management 7.1 (2016): 41-49.
24. Singh, Neha, and Kirti Tyagi. Ranking of services for reliability estimation of SOA system using fuzzy multicriteria analysis with the similarity-based approach. International Journal of System Assurance Engineering and Management 8.1 (2017): 317-326.
25. Abreu, Fernando B., Carapuca, Rogerio, Candidate metrics object-oriented software within taxonomy Framework. Journal of systems software 1994.
26. Chidamber, Shyam, A metrics suite for object-oriented design. 1993.
27. Li. Wei., Hemery Maintenance Metrics for the object-oriented paradigm First international software metrics Symposium. IEEE computer science press 1993.
28. Abreu, Fermando B. The Mood Set, Proc. ECOOP 95 workshop on metrics 1995
29. Shatnawi R, A quantitative investigation of the acceptable risk levels of object-oriented metrics in open –source system IEEE Transactions 2010.
30. Amargo Cruz Ana Erika, Chidamber & Kemrer suite of metrics Japan Advanced Institute of science and Technology School of information, May 2008.
31. Rosenberg, L. H. and Hyatt, L., Applying and interpreting object-oriented metrics, in Proceedings of Software Technology Conference, Utah, April 1998.
32. Rosenberg, L. H. and Lawrence, E. H., Software Quality Metrics for Object-Oriented Environments, Unisys Technology Conference, Virginia, 1996.
33. Chidamber, S. R. and Kemerer, C. F., A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering, vol. 20, pp. 476-493, 1994.
34. Succi, G, Pedrycz, W., Stefanvic, M., and Miller, J., Practical assessment of the models for identification of defect-prone classes in object-oriented commercial systems using design metrics, The Journal of Systems and Software, vol. 65, pp. 1–12, 2003.
35. Basili, V. L., Brianc, L., and Melo., W. L., A Validation of Object-Oriented Metrics as Quality Indicators, IEEE Transactions Software Engineering, vol. 22, pp. 751-761, 1996.
36. McQuillan, J. A. and Power, J. F., On the application of software metrics to UML models, Springer Lecture Notes in Computer Science, vol. 4364, pp. 217-226. 2007.
37. Richard W. Selby, Enabling Reuse-Based Software Development of Large-Scale Systems, IEEE Transaction of Software Engineering, Vol. 31, No. 6, PP. 495-510, Jun 2005.
38. Parvinder Singh Sandhu and Hardeep Singh, Automatic Reusability Appraisal of Software Components using Neuro-Fuzzy Approach, International Journal Of Information Technology, vol. 3, no. 3, 2006, pp. 209-214.
39. Parvinder S. Sandhu Pavel Blecharz and Hardeep Singh, A Taguchi Approach to Investigate Impact of Factors for Reusability of Software Components, World Academy of Science, Engineering and Technology, pp.135-140, Sep 2007.
40. Gui Gui and Paul D. Scott, New coupling and cohesion Metrics for Evaluation of Software Component Reusability, in Proc. ICYCS, 2008, pp.1181- 1186.
41. Parvinder S. Sandhu, Harpreet Kaur and Amanpreet Singh, Modeling of Reusability of Object-Oriented Software System, World Academy of Science, Engineering and Technology Issue. 30, pp. 162-165, Aug 2009.
42. Sonia Manhas, Rajeev Vashisht, Parvinder S. Sandhu and Nirvair Neeru, Reusability Evaluation Model for Procedure Based Software Systems, International Journal of Computer and Electrical Engineering, Vol.2, No.6, pp. 1107-1110, Dec 2010.
43. Nasib S. Gill and Sunil Sikka, Inheritance Hierarchy Based Reuse & Reusability Metrics in OOSD, International Journal on Computer Science and Engineering (IJCSE), Vol. 3 No. 6, pp. 2300-2309, June 2011.
44. Fazal-e-Amin, Ahmad Kamil Mahmood and Alan Oxley, Reusability Assessment of Open Source Components for Software Product Lines, International Journal on New Computer Architectures and Their Applications (IJNCAA), 1(3), pp. 519-533, 2011.
45. Ajay Kumar, Measuring Software Reusability using SVM based Classifier Approach, International Journal of Information Technology and Knowledge Management., Vol. 5, No. 1, pp.205-209, Jan 2012.
46. Neha Goyal, Er. Deepali Gupta, Reusability Calculation of Object-Oriented Software Model by Analyzing CK Metric, International Journal of Advanced Research in Computer Engineering & Technology, Volume 3 Issue 7, July 2014
47. Padhy, N., Singh, R. P., & Satapathy, S. C. (2018). Software reusability metrics estimation: Algorithms, models and optimization techniques. Computers & Electrical Engineering, 69, 653–668. doi:10.1016/j.compeleceng.2017.11.022
48. Padhy, Neelamadhab, Rasmita Panigrahi, and K. Neeraja. Threshold estimation from software metrics by using evolutionary techniques and its proposed algorithms, models. *Evolutionary Intelligence* (2019): 1-15.

49. Padhy, N., Singh, R. P., & Satapathy, S. C. (2017). Enhanced evolutionary computing based artificial intelligence model for web-solutions software reusability estimation. Cluster Computing.doi:10.1007/s10586-017-1558-0
50. Hudaib, Amjad, Ammar Huneiti, and Islam Othman. Software Reusability classification and predication using self-organizing map (SOM). Communications and Network 8.03 (2016): 179.
51. M. Shaw, R. DeLine, D.V. Klein, T. L. Ross, D.M. Young, G. Zelesnik, Abstractions for Software Architecture and Tools to Support Then, IEEE Transactions on Software Engineering, April 1995.
52. . Sandhu, P., & Singh, H. (2005). A Critical Suggestive Evaluation of CK metric. PACIS 2005 Proceedings, 16.
53. Coleman, C. Principal Components of Orthogonal Object-Oriented Metrics. White Paper Analyzing Results of NASA Object-Oriented Data (323-08-14), October 2001. Postal addresses István Siket. In Department of Software Engineering University of Szeged H-6720 Szeged, Árpád tér 2, Hungary Rudolf Ferenc Department of Software Engineering University of Szeged.