Reviewing the Significance of Software Metrics for Ensuring Design Reusability in Software Engineering

P. Mangayarkarasi Palaniswamy Velumani
New Horizon College of Engineering
7 PUBLICATIONS **47** CITATIONS

R.Selvarani Rangasamy
Alliance University
66 PUBLICATIONS **296** CITATIONS

ISSN:2249-5789

Reviewing the Significance of Software Metrics for Ensuring Design Reusability in Software Engineering

P. Mangayarkarasi
Research Scholar
Visvesvaraya Technological University Belgaum, India
E-Mail: mangaivelu18@gmail.com

Dr. R. Selvarani Professor CS/IT,
Alliance college of Engineering and Design,
Alliance University, Bangalore
E-Mail: Selvarani.riic@gmail.com

***Abstract*— with the rise of high competition to retain maximum quality cost effective software application, the significance of software engineering are enhancing in quite faster pace. The field of software development is increasingly giving more emphasis on the object oriented design as well as software metrics as essential method to ensure the quality of software. There has been a quite abundant of studies conducted in the past addressing to the issues of object-oriented development, however, no studies were found effectively for design reusability from software engineering viewpoint. This paper therefore discusses about the essentials of design reusability and its significant charecteristics, which has potential features for cutting the cost of development. The paper also discusses about the most frequently used software metrics till date as well as less -used software metrics. Finally, the paper discusses about the open issues from the studies.**

Keywords: Component, Design Reusability, Design Pattern, Software Metrics, CK Metrics, MOOD, etc

I. INTRODUCTION

In the area of software development methodologies, object oriented designs are considered as one of the significant attributes to measure the quality aspects of the software [1]. It was also seen that software projects of small/large scale uses object oriented design methodologies for any software development organization. Hence, object oriented designs can be considered as a degree using which the system objects can posses the specific attributes as well as required charecteristics. The prime reason behind this large scale adoption is that object orient methodologies basically visualize the problems and tends to give solution based on all the micro and macro level problems in terms of objects thereby ensuring better adaptability, reliability, flexibility, and also reusability [2]. At present, the software metrics are used by the engineers to evaluate the required resources and design component for a particular software project. Hence, the significance of software metric is that it provides a better platform to evaluate the design pattern as well as assist in testing the application in quantitative manner. Such testing assists in ensuring better software reliability too. Basically, when a company gets a new requirement for any clients, they formulate a design of that requirement, where the confirmed designed by technical architect goes for production. Once the code is designed, the applications are sold to customers. However, it is also unethical to reuse the code of previous clients for the purpose of developing new application for new clients [3][4]. Hence, a production team has to go for new development from scratch, which not only takes their effort, but also time and money. Majority of the large scale organization now-a-days uses the design pattern where the design patterns are subjected for reuse without any ethical issues. However, a question also arise that how much proportion of the old design for new requirements can be reused? Answering this question requires to be seen from the evidences from the literature survey, where various other techniques pertaining to design patterns of object oriented methodologies needs to be studied. The literature needs to be also analyzed to see how many of such software metrics existing in past and present that permits design reusability.

The prime purpose of design reuse [5][6] is to provide the assistance to the developers to use it for the new production, which drastically cut down the cost of new development from the scratch. However, when software engineers are working on design reusability, it is essential that the existing design patterns to be optimally reused for the existing client as well as for the future clients too. Adoption of design reuse also ensure the production and delivery process to meet on time (or sometimes before time), which gives lots of scope to the development team to ensure the quality of their production. Design reuse doesn't means that 100% of the design could be reused. It means that possible 40% of the prior design could be reused while it calls for fresh 60% of the designs should be developed from base for meeting a particular requirement of clients. Hence, the design team needs to ensure that the new 60% design should not be focused only for the existing client, but even it should have minimum proportion of design reusability for its future clients too. However, it is not so easy to do, as client's upcoming requirements cannot be predicted. Section 2 discusses about the fundamentals of design reusability along with its considered attributes. Section 3 discusses about the desired charecteristics of the design reusability. Section 4 discusses about the unconventional object oriented metrics and Section 5 discusses about the conventional object oriented metrics. Section 6 highlights about the most frequently adopted software metrics called as CK metrics, while Section 7 discusses about some of the significant studies done most recently. Finally, Section 8 discusses about an open issue and followed by summarization of the paper in Section 9 as conclusion.

II. DESIGN REUSABILITY

Design Reusability is the one of the critical requirements for all the companies who is into product development [7][8]. Design reusability can be basically used as a framework in design patterns as an extent of ease with which one can deploy priorly designed frameworks in the novel applications. The design aspects can be reused in multiple different tasks on the applications, which can be reused in the same system at the multiple different level or it may be reused in many other applications too. The outcome of the design reusability is basically a design where the cost of new development is lowered subjected the reusability factor of the resultant design is higher. The development of new design is basically not targeted only for the current consumer but also for future consumers and requirements. Therefore, it is important that the design architect should concentrate on the needs of large proportion of customer rather than the existing customers only. Hence, the importance is given on the design reusability for existing as well as future clients too. Hence, anticipated return-on-investment is proportionately high when the development companies successfully implement design reusability. Therefore for effective design reusability, following attributes should be considered:

- The resultant design is anticipated to be highly generic and should encapsulate the existing as well as the upcoming need of anticipated customers. The system must also consider the unique designing aspects apart from the reusability part.
- For better risk management, design should be also focused on unknown need of customers (to avoid requirement volatility issues)
- Emphasis should be given on the design interface contract to be more simplified and benchmarked for its extendibility to multiple different customers in future (or in present as well).
- Design comprehension and compatibility should be clear and precise for the customers to adopt and operate easily.
- Operation and associated functionality of the anticipated design should be highly enriched.
- The new design should have better exceptional handling.
- The design portability should have better scope.
- The reused design should be free from other design process.

20 years back, design reusability was not emphasized much, but with the upcoming cut-edge competition for retaining maximum number of clients with cost effective human resource, design reusability has found a peak position of importance in every software development companies. It is one of the media to foresight the future needs of anticipated customers reducing the cost of new development. Hence, there are multiple benefits of adopting design reusability e.g. i) minimization of design duplication, ii) minimization of development cost and duration, iii) Maximization of return of investment and enhancement of productivity, iv) Non-trivial maintenance, v) increases reliability and reduces risk.

III. Characteristics of Design Reusability

For making the design as a reusable one, the emphasis should be given to both design and quality characteristics. At the production stage the design should adhere to other design characteristics which enhance design reusability. The essential charecteristics of design reusability are as follows [9]:

- **Loose Coupling**: Design loose coupling enhances the design reusability. The lower the dependency with other design, the more easily it can be reused.
- **Composability**: Design composability is the key principle for reusability. The composable design can easily integrate with other design. Therefore the design composability offers higher degree of design reusability.
- **Autonomy**: The reusable design should be independent. If the design is independent from other design and business logic and self governance, then the design will be more reusable.
- **Abstraction**: Design abstraction hides the unnecessary information from the design consumers. Also it reduces the needless coupling between the design consumer and design provider thereby increases the design reusability.
- **Statelessness**: Statelessness encourages design reusability. Lesser the amount of state management responsibilities increases its scalability and availability which are the required qualities to enhance design reusability.
- **Discoverability**: Design discoverability promotes design reusability. If and only if the design consumer can easily find the required service, the design can be more reusable.
- **Granularity**: The design granularity may be fine grained or coarse grained. Depending upon the type of design, the granularity level may vary. The correct granularity level of the design enhances the design re-use.

There is couple of studies done in the past for understanding about pros and cons of design reusability, where effectively majority of the studies has focused on the object oriented software metrics. The next section will discuss about some of the studied software metrics in object oriented development.

IV. Unconventional Object Oriented metrics

This section discusses about the unconventional object oriented metrics that are considered in the development of software projects. The unconventional terms is coined as such metrics were although formulated was found very less to be adopted in majority of the studies related to objected oriented development.

- **Chen Metrics:** Chen et al. [10] proposed software metrics, through which it can define "What is the behavior of the metrics in object-oriented design". They may be described all of the behaviors like: (i) CCM (Class Coupling Metric), (ii) OXM (Operating Complexity Metric), (iii) OACM (Operating Argument Complexity Metric), (iv) ACM (Attribute Complexity Metric), (v) OCM (Operating Coupling Metric), (vi) CM (Cohesion Metric), (vii) CHM (Class Hierarchy of Method) and (viii) RM (Reuse Metric). Metrics (i) and (iii) are very subjective in nature, Metrics (iv) and metric (vii) mostly involve the count of features; and metric (viii) is a Boolean (0 or 1) indicator metric. Therefore, all of the terminologies in object oriented language, consider as the basic components of the paradigm are objects, classes, attributes, inheritance, method, and message passing. They proposed all of that each object oriented metrics concept implies a programming behavior.

- **Morris Metrics:** Morris et al. [11] proposed a metrics suite for the object-oriented metrics systems and they define the system in the form of the tree structure and the following are the Morris's complexity and cohesion metrics. Morris defined the complexity of the object oriented system in the form of the depth of the tree. Depth of the tree measures the number of the sub-nodes of the tree. The more the number of sub nodes of tree the more complex the system. So, complexity of an object is equal to the depth of tree or total number of sub nodes.
- **Lorenz & Kidd Metrics:** Lorenz & Kidd [12] proposed a set of metrics that can be grouped in four categories are size, inheritance, internal and external. Size oriented metrics for object oriented class may be focused on count of the metrics, operations and attributes of an individual class and average value of object-oriented software as a whole. Inheritance based metrics is totally concentrated in which operations that are reused through the class hierarchy. Metrics for the class intervals are totally oriented towards the cohesion, while the external metrics were used to examine and reuse. It divide the class based metrics into the broad categories like size, internal, external inheritance and the main metrics which are focused on the size and complexity are class size (CS), Number of operations overridden by a subclass (NOO), Number of operations added by a subclass (NOA), Specialization index (SI), Average operation size (OS), Operation complexity (OC), Average number of parameters per operation (NP).

V. CONVENTIONAL OBJECT ORIENTED METRICS

This section discusses in brief about the conventional object oriented metrics. The term conventional object oriented metric is coined as following metrics are found to used in majority of the research work in past decade.

- **MOOD**: Abreu et al. [13] defined MOOD (Metrics for Object Oriented Design) metrics. They evaluated that how OO design mechanisms like inheritance, polymorphism, information hiding and coupling can make an influence on quality characteristics like defect density (a reliability measure) and rework (a maintainability measure). They also derived certain criteria like metrics should be formally defined, dimensionless, obtainable early, down-scalable, easily computable. They should be language and size independent. MOOD metrics refers to a basic structural mechanism of the object-oriented paradigm like encapsulation (MHF and AHF), inheritance (MIF and AIF), polymorphism (PF) and message passing (CF). MOOD metrics are based on set theory and includes simple mathematics. These are applicable as soon as a preliminary design is available so the flaws can be detected in the early phase. Subjectivity is avoided as these are formally defined.
- **QMOOD**: QMOOD (Quality Model for Object Oriented Design) was proposed by Bansiya and Davis [14]. It is the comprehensive model that assesses quality attributes like reusability, functionality, effectiveness, understandability, extendibility, flexibility. There are four levels (L1 through L4) and three mappings to connect these levels in QMOOD. The four levels are: A. Design Quality Attributes. B. Object oriented design Properties. C. Object oriented design Metrics. D. Object oriented design Components

However, some researchers [15] who have deeply evaluated MOOD have contradicted that the majority of the metrics involved in MOOD has high range of software defect. However, the author has also commented that it is not necessary to point out the demerits of MOOD or QMOOD as they have other potential advantage features too. Hence, last half decade has witnessed frequent adoption of CK metric (although it has been evolved in 1994). According to various researchers, CK metrics is better replacement of other conventional and unconventional software metrics for object oriented development. The next section will discuss about CK metrics and its associated working principles.

VI. CK-METRICS

The pioneering of the potential software metrics was done by Chidamber and Kemerer [16] in 1994 who have introduced a standard software metrics for object oriented programs. CK metrics or Chidamber and Kemerer metrics plays a significant role to know the design aspects of the software and to enhance the quality of software [16]. Previous studies [16][17] show that the majority of the metrics suites are designed based upon the original CK metrics suite. The prime purpose of CK metric is to furnish a detailed evaluation of the cumulative quality of the software programs for all the level of class. The metrics are associated with each small segment of the software providing in-depth information of the software and its quality. The CK metrics suite proposes class-based six metrics that assess different characteristic of object oriented programs, having the following metrics: (i) Weighted Methods per Class (WMC), (ii) Response for a Class (RFC), (iii) Lack of Cohesion of Methods (LCOM), (iv) Depth of Inheritance Tree (DIT), (v) Number of Children (NOC), and (VI) Coupling between Object classes (CBO). Though the original suite of CK design metrics has six metrics, the present paper will consider five metrics. The five metrics of CK Suite are described as follows:

1. **Weighted Methods per Class**: It is a number of an effective method to that are implemented inside a class where class may possess bigger quantity of methods specific to applications [18]. This metric minimizes the dependability and understandability.
2. **Response for a Class**: This metric is a number of cumulative methods inside a set that can be called upon in response to the message sent to an object for carrying out a specific task [18].
3. **Depth of Inheritance Tree**: One of the frequently used metrics, it estimates the extent of depth in the hierarchy of any class. It also evaluates maintainability and reusability.
4. **Number of Children**: It is a measure of the number of classes associated with a specified class with an

aid of an inheritance relationship. A class having many children is a bad class with a bad design [19].

5. **Coupling between Object classes**: It is defined as the number of all the other set of classes to which it is coupled. CBO is beneficial in judging the complexity of testing and reusability [16]. Among the proposed CK metrics, the effective metrics are WMC, RFC, DIT, NOC and CBO.
6. **Lack of Cohesion of Methods (LCOM):** LCOM is the difference between the number of methods whose similarity is zero and the number of methods whose similarity is not zero. The similarity of two methods is the number of attributes used in common. However, Basili et al. [20], Briand et al. [21] and Kaur in [22] noted problems in the LCOM metrics, a value of zero of LCOM is not an evidence of cohesiveness and also very high value of LCOM does not depict any inference. LCOM metric makes it difficult, if not impossible, to define a unit and to measure quality. LCOM does not quantify quality accurately and is not a good measure.

Usually it was seen that approaches for designing software metric frequently use single snapshot of a software project. Evaluating a project over a longer time-frame permits the consideration of other software quality facets, such as reuse and maintainability. Across a wide variety of reported results from using Object Oriented (OO) metrics in industrial settings and using data from an assortment of countries and applications, we can make several observations:

- OO metrics have been successfully applied in various domains and programming languages in countries worldwide.
- They have consistently demonstrated relationships to quality factors such as cost, defects, reuse, and maintainability—relationships that go above and beyond that of size.
- Inheritance (measured by DIT or NOC) is apparently used only sparingly in practical OO applications, and thus its relationship to project outcomes is less certain.

Here, the user is a software engineer or developer. Hence, internal usability metrics are used for predicting the extent, to which the software in question can be understood, learned, operated, and is attractive and compliant with usability regulations by integrating it with a larger software system. Understandability is defined as the attribute of software that bears on the users' efforts in recognizing the logical concept and its applicability. Learnability is defined as the attribute of software that bears on the users' efforts for learning its application. The operability is defined as the attribute of software that bears on the users' efforts for operation and operation control. Attractiveness is defined as the attributes of software that bear on the capability of the software product to be attractive to the user. Table 1 shows the five CK metric with respect to understandability, Learnability, Operability, and attractiveness measures. It also shows that CBO, and RFC are not addressed by Learnability and operability. Review of some significant approach is stated below:

Table1 Facts of CK Metrics [23]

CK Metrics	Understandability	Learn ability	Operability	Attractiveness
WMC-Weighted Methods per class	√	√	√	√
DIT-Depth of inheritance Tree	√	√	√	√
NOC-Number of Children	√	√	√	√
CBO-Coupling Between Object Classes	√	-	-	√
RFC-Response Set for Class	√	-	-	√
LCOM-Lack of Cohesion in methods	√	-	-	√

VII. Recent Studies

This section discusses about the most significant studies captured in the recent past related to the issues of design reusability using software metrics in the area of software engineering. Nair and Selvarani [24] proposed a framework with a capability to forecast the reusability index considering three metrics in Chidamber and Kemerer metrics viz.: DIT, RFC and WMC. They exposed the strong relationship that exists between the design parameters and reusability factors in developing a reusability estimation model. Nair and Selvarani [23] carried out a complete analysis of the relationships that exist between internal quality attributes in terms of the complete suite of Chidamber and Kemerer metrics and the reusability index of software systems. The authors presented a new regression technique for the purpose of mapping the association between reusability and design metrics. Selvarani [25] presented an empirical evaluation of the Chidamber and Kemerer metrics for assessing prediction capability using data driven techniques for mitigating defects in software. The author has carried out the investigation considering Weighted Methods per Class mainly. Selvarani [26] presented an extensive evaluation framework for assessing the impact of defects in software using data driven techniques. The study was conducted in the direction of defect evaluation in the design stage of Object Oriented programs. The final outcome of the study shows better efficiency in the existing development lifecycle of softwares. Neha Budhija et al. [27] proposed an approach for identifying and qualifying reusable software components with a few metrics like, index of coupling, inheritance, external dependency, and polymorphism. Kaur et al. [28] analyzed the standard MOOD metrics along with assessing the Chidamber and Kemerer metrics. The study presented a standard reusability metric model with higher dimensional scope in metrics related to object oriented programs. Gill and Sikka [29] presented a framework of reusability and discussed Object Oriented programs with a viewpoint of evaluating inheritance hierarchy, for which purpose, the authors developed 5 novel metrics. Using the reuse metrics, the authors performed precise classification. Goel and Bhatia [30] elaborated the design of the CK metric suite as well as performed an analysis on those metrics for the purpose that these metrics should highlight precise results for object oriented systems. Subramanyam and Krishnan [31] provided empirical evidence that supports to solve the complexities in the object oriented programs for identifying defects. The primary finding of the study states that the Chidamber and Kemerer metric support the flexibility for amendments of mitigating defects in Object Oriented programming

VIII. Open Issues

While performing random exploration for contributory work in the same field, it was seen that the author has a higher set of contributory work in the same field in which our research lies. After Reading the article '*A Critical Suggestive Evaluation of CK Metric*' [32], we came to know about the validation criterion for CK Metric Suit that has 9 properties to measure (Non-Coarseness, Granularity, Design Details, Monotonicity, Non-Equivalence of interaction, Non-Uniqueness, Permutation of elements, Renaming, Interaction increases complexity). The author has also discussed demerits of CK metrics as tabulated below:

- WMC-Weighted Methods per class:
  - WMC break an elementary rule of measurement theory.
  - This is also not clear whether the inherited method is to be counted in base class (which defines it), in derived classes or in both.
- DIT-Depth of inheritance Tree:
  - There is the inconsistency in the theoretical basis and definition of the metric in case of multiple inheritances.
  - Deeper Inheritance produces hindrances in maintenance. On the other hand it states that it is better to have Depth than breadth in the Inheritance Hierarchy Hence there is contradiction in the statements of DIT metric.
- NOC-Number of Children:
  - The definition of NOC metric gives the distorted view of the system as it counts only the immediate sub-classes instead of all the descendants of the class
- CBO-Coupling Between Object Classes:
  - -As Coupling between Object classes increases, reusability decreases and it becomes harder to modify and test the software system.
  - For most authors coupling is reuse, which raises ambiguity.
  - Chidamber and Kermerer state that their definition of coupling also applies to coupling due to inheritance, but do not make it clear if all ancestors are involuntarily coupled or if the measured class has to explicitly access a field or method in an ancestor class for it to count.
- RFC-Response Set for Class:
  - Chidamber and Kermerer recommended only one level of nesting during the collection of data for calculating RFC. This gives incomplete and ambiguous approach as in real programming practice there exists "Deeply nested call-backs" that are not considered here.
- LCOM-Lack of Cohesion in Methods:
  - The definition of CK metric for LCOM is not able to distinguish the more cohesive class from the less ones. This is simple violation of the basic axiom of measurement theory, which tells that a measure should be able to distinguish two dissimilar entities. So this deficiency offends the purpose of metric.

Hence, it can be seen that even frequently adopted CK metrics is not without flaws and hence, it can be concluded that CK Metrics should be thoroughly amended for making it eligible for incorporating design reusability in software development methodologies. Hence, as a research gap, there are no studies being explored till date to ensure design reusability in software engineering.

IX. Conclusion

This paper discusses the evolution of design reusability, as well as focus on conventional and unconventional software metrics. The paper contributes to precise understanding of the design reusability and its associated feature. It also states that some of the metrics s like Chen's metrics, Morris Metrics, and Lorenz & Kidd are less used metrics, where MOOD and CK metrics were found very high. While visualizing the most recent significant studies, it was found that majority of the researchers are more inclined towards CK metrics. However, as a research gap, none of the studies were explored to address the design reusability using CK metrics.. Therefore, our future work formulates a mathematical model using CK metrics that can show the significance of adoption of design reusability from software engineering viewpoint.

References

[1] S. K. Dubey, A. Rana, A Comprehensive Assessment of Object-Oriented Software Systems Using Metrics Approach, International Journal on Computer Science and Engineering Vol. 02, No. 08, 2010, 2726-2730

[2] N. Mohammed, A. Govardhan, Comparison between Traditional Approach and Object-Oriented Approach in Software Engineering Development, International Journal of Advanced Computer Science and Applications, Vol. 2, No. 6, 2011

[3] B.Jalender, A.Govardhan, P.Premchand, Designing code level reusable software components, International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.1, January 2012

[4] P.Niranjan, P.Shireesha, M.Venugopal Reddy, Development of Reuse Repository and Software Component Performance Analysis, International Journal of Application or Innovation in Engineering & Management, Volume 2, Issue 6, June 2013

[5] N Md Jubair Basha, and Salman Abdul Moiz, A Framework Studio for Component Reusability, CoRR abs/1202.5609, 2012

[6] Florinda Imeri, Ljupcho Antovski, An Analytical View on the Software Reuse, I ICT Innovations, ISSN 1857-7288, 2012

[7] R.L. Glass, Frequently Forgotten Fundamental Facts aboutSoftware Engineering, IEEE SOFTWARE May/June 2001

[8] P. T. Devanbu, D.E. Perry, Jeffrey S. Poulin, Guest Editors' Introduction: Next Generation Software Reuse, IEEE Transactions on Software Engineering, VOL. 26, NO. 5, MAY 2000

[9] Domingue, John; Gonzalez-Cabero, Rafael and Fensel, Dieter (2008). SOA4All, enabling the SOA revolution on a world wide scale. In:

Second IEEE International Conference on Semantic Computing (ICSC 2008), 4-7 August 2008, Santa Clara, CA, USA

[10] Chen, J-Y., Lum, J-F.: "A New Metrics for Object-Oriented Design.", Information of Software Technology 35,4(April 1993) :232-240

[11] K. Morris, "Metrics for Object-oriented Software Development Environments," Masters Thesis, MIT, 1989..

[12] M. Lorenz, J. Kidd, "Object Oriented Software Metrics", Prentice Hall, NJ, (1994).

[13] B. F. Abreu: "Design metrics for OO software system", ECOOP'95, Quantitative Methods Workshop, 1995.

[14] J. Bansiya, C. G. Davis, "A Hierarchical Model for Object-Oriented Design Quality Assessment", IEEE Transactions on Software Engineering, 28, (1), (2002), 4–17.

[15] T. Mayer & T. Hall, Measuring OO Systems: A Critical Analysis of the MOOD Metrics, Journal of Software Quality Control, Vol.8, iss.2, 1999

[16] S.R. Chidamber, C.F. Kemerer. 1994. A Metric Suite for Object Oriented Design, IEEE Transactions on Software Engineering, Vol.20, No.6

[17] S. Singh, P. Singh, N. Mohan, P.S. Sandhu. 2012. Logistic Model Trees based Approach for Prediction of Reusability of Object Oriented Software Components. International Journal of Research in Engineering and Technology, Vol. 1, No. 3

[18] M. Sharma, G. Singh, A. Arora, P. Kaur. 2012. A Comparative Study of Static Object Oriented Metrics, International Journal of Advancements in Technology, Vol. 3, No.1

[19] A. Chatzigeorgiou. 2003. Mathematical Assessment of Object-Oriented Design Quality. IEEE Transactions on Software Engineering, Vol. 29, No. 11

[20] Basili, V.L., Briand, L., and Melo, W.L., A Validation of Object-Oriented Metrics as Quality Indicators, IEEE Transactions Software Engineering, Vol. 22, No. 10, 1996

[21] Briand, L.C., Wust, J., Daly, J.W., and Porter, D.V., Exploring the Relationship Between design Measures and Software Quality in Object-Oriented Systems, Journal Systems and Software, Vol. 51, No. 3, 2000

[22] Kaur, S., Singh, S., Kaur, H., A Quantitative Investigation Of Software Metrics Threshold Values At Acceptable Risk Level, International Journal of Engineering Research & Technology, Vol. 2 Issue 3, March - 2013

[23] R. Selvarani, T.R.Gopalakrishnan Nair. 2009. Software Reusability Estimation Model Using Metrics Governing Design Architecture, International Book: "Knowledge Engineering for Software Development Cycles: Support Technologies and Applications", Engineering Science Reference, IGI Publishing, USA, DOI: 10.4018/978-1-60960-509-4.ch011

[24] T.R.Gopalakrishnan Nair and R. Selvarani. 2010. Estimation of Software Reusability: An Engineering Approach, Association for Computing Machinery (ACM) – SIGSOFT, USA, Vol.35, Iss.1

[25] R. Selvarani. 2010. Software Metrics Evaluation Based on Entropy, Handbook of Research on Software Engineering and Productivity Technologies: Implications of Globalization, DOI: 10.4018/978-1-60566-731-7.ch011, 2010

[26] T.R. Gopalakrishnan Nair, R. Selvarani. 2012. Defect proneness estimation and feedback approach for software design quality improvement, Information and Software Technology, Elsevier, vol.54, pp. 274–285

[27] N. Budhija, B. Singh, S.P. Ahuja. 2013. Detection of Reusable Components in object Oriented Programming Using Quality Metrics. International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, Iss.1.

[28] A. Kaur, S. Singh, K.S. Kahlon, P.S. Sandhu. 2010. Empirical Analysis of CK & MOOD Metric Suite. International Journal of Innovation, Management and Technology, Vol. 1, No. 5.

[29] N.S. Gill, S. Sikka. 2011. Inheritance Hierarchy Based Reuse & Reusability Metrics in OOSD, International Journal on Computer Science and Engineering, Vol. 3, No. 6.

[30] B.M. Goel, P.K. Bhatia. 2012. Analysis of Reusability of Object-Oriented System using CK Metrics, International Journal of Computer Applications, Vol.60, No.10, pp.0975 – 8887.

[31] R. Subramanyam, M.S. Krishnan. 2003. Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects. IEEE Transactions on Software Engineering, Vol. 29, No. 4.

[32] P. Sandhu, H. Singh, 'A Critical Suggestive Evaluation of CK Metric', PACIS, 2005