INTERNATIONAL JOURNAL OF SCIENCE
ISSN: 2455-0108 | WWW.IJOSCIENCE.COM
VOLUME II ISSUE II APRIL 2016

A BRIEF REVIEW OF SECURITY METRICS FOR OBJECT-ORIENTED CLASS DESIGNS

Mona Vishwakarma[1], Vaibhav Udgir[2]

[1]PG Scholar, College name, NRI Institute of Information Science and Technology, Bhopal, India
[2] Professor, NRI Institute of Information Science and Technology, Bhopal, India

Abstract:
Measuring quality characteristics of item situated plans (e.g. viability and execution) has been secured by various studies. Be that as it may, these concentrates on have not considered security as much as other quality properties. Additionally, most security studies center at the level of individual system explanations. This approach makes it hard and costly to find and fix vulnerabilities brought about by outline blunders. These metrics enable designers to get associate degreed fix security vulnerabilities at an early stage, and facilitate compare the safety of varied different styles. particularly, we tend to review pervious security metrics to liveknowledge Encapsulation (accessibility) and Cohesion (interactions) of a given object-oriented category from the purpose of read of potential info flow
and conjointly discuss concerning their options and downsides.

***Index Terms*— Channel estimation, Multiple Input Multiple Output, Least Mean Squares, Normalized LMS, Variable Step Size LMS, Recursive Least Squares, Least Mean-Squares Newton Algorithm, Mean square error.**

I. INTRODUCTION

The software industry having lack of standard metrics and measurement. Software security is a major activity in the software industry. It has been reported that cost and effort spent on software security is very high, approximately between 65% to 70% of total software development and support efforts [1]. Software reengineering, recently, have been advocated as a means of reducing security costs [9]. Slightly short of software metric has multiple definition and ambiguous rules**.** It is tough to discover vulnerabilities within the operational stage of code, as a result of the protection concern aren'taddressed or illustrious sufficiently early throughout code development. Accessibility (data encapsulation: the mechanism that binds along the code and therefore the information it manipulates, and keeps each safe from outside interference and misuse) and interaction (cohesion: a live of however powerfully connected or targeted the responsibilities of one module are) i.e. connected code metrics will be measured throughout the sooner phases of code development. To satisfy security demand, it's necessary to guard information from unauthorized speech act of data and alteration of data.

Taking security early phase of a system development should have an impact on reducing many software vulnerabilities. Software vulnerability is an instance of a [fault] in the specification, development, or configuration of software such that its execution can violate an [implicit or explicit] security policy [7].Software security is the ability to defend attacker's exploitation of software problems by building software to be secure throughout the whole development life cycle [4]. Object oriented class design is becoming more famous in software development and object oriented design metrics is an important part of software development environment. This study is focuses on a set of object oriented security design metrics that can be used to measure the security and quality of an object oriented class design. The metric for object oriented class design focus on measurement that are applied to object oriented class and design characteristics. These measurement permits designers to access the software design in early phase of development, making changes that will reduce the complexity and improve the continuing capability of the design. The object oriented model closely represents the problem domain, which makes it easier to produce and understand design. It is also believed that object oriented design will encourage more re-use, i.e., new application can use existing modules more efficiently and effectively, thereby, reducing development cost and time [6].Security measurements have been defined to assess security at the level of implementation code [3]. This paper shows the study of different security design metrics. These Metrics permit designer of system to get and fix the protection of varied different of sophistication styles. we tend to conjointly mention the analysis of those metrics. These observations show that security style metrics will be used as early indicators of vulnerability in package. within the next session we offer a general background regarding metrics and package measures and existing package measurement metrics.

1.1. *Need of Software Metrics*

The use of software metrics is for developing quality software. Software metrics measures of some piece of property of software, or software specifications. Software metrics are measures of the attributes of the software products and

processes. Software metrics are measures that could be used to measure different characteristics of a software system.

II. RELATED WORK

Most present studies on programming security concede that there is no such thing as a totally secure system, be that as it may, there are in any case different methods for diminishing security dangers and vulnerabilities [8] [9]. One of these is the requirement of security in the execution.
A few activities have been directed to research data course through PC program code. This has been considered utilizing a few methodologies, including sort investigation [10] and information/control-stream examination [11].

Another methodology is to implement security at right on time periods of the product improvement lifecycle, for example, at the outline stage. One of the soonest concentrates on around there was the improvement of programming security outline standards by Saltzer and Schroeder [12]. These standards were expected as direction to create secure frameworks, chiefly working frameworks.
Cleric's [13] and McGraw's [14] writings distinguished a few comparable security plan standards. Nonetheless, these standards were not fit for evaluating the security levels of projects. Accordingly, there is a requirement for security measurements in view of these standards to unbiasedly measure the security of a given system specifically from its configuration curios. Characterizing programming security measurements is another method for decreasing project security dangers and vulnerabilities. An existing methodology which is utilized by software engineers to evaluate the level of security of given system code is taking into account the distinguishing proof of vulnerabilities [15] [9].
A study directed by Chowdhury et al. [3] characterized a number of security measurements that evaluate the security of a given project in light of code reviews. These measurements require full usage of the framework to evaluate its security. This methodology makes it difficult to settle security mistakes at the outline time and is costly in terms of time and assets. Furthermore, measuring the security of the framework's engineering has been finished by Manadhata et al. [2].
This study concentrated on the framework's 'assault surface'. Additionally, a study that characterized outline measurements which measure certain product quality properties was directed by Bansiya [16]. He recognized a methodology to enhance the Quality Model for Object-Oriented Outline (QMOOD) [5]. The model intends to gauge the nature of different article arranged configuration qualities for example; reusability, adaptability, and usefulness based on their pertinence to certain quality configuration properties (e.g. deliberation, union, and coupling). Despite the fact that the study secured most plan quality traits, it did not consider security. Since security is a quality prerequisite [1], building up various security measurements in view of the QMOOD quality configuration properties is the best choice for planning measurements. Characterizing an arrangement of measurements which assesses the security of a given project in light of its outline antiques instead of its source code would diminish the expense of settling security outline vulnerabilities by identifying these vulnerabilities at an early stage.
Bandar Alshammari, Colin Fidge and Diane Corney [14] defined a number of security metrics for object-oriented designs. These metrics are easy to capture and apply once a given class is designed and annotated using UMLsec and SPARK's annotations. The metrics not only allow designers to define the most secure design but they can also give indications of where any potential vulnerability occurs. They differ from code level metrics as they are easier to capture and don't require the software to be implemented. We have also shown how to directly compare the metrics results for various alternative designs and thus help choose the design which best satisfies a certain security design principle. The defined approach can also make it easier for systems designers to choose which refactoring methods to use to satisfy a certain security design principle.
Software metrics can be used to find out the properties of the software that we are developing and predict the needed effort and development period. "LOC (Lines of Code)" is one of the most primitive and oldest metrics. In the beginning of 1990s, Chidamber and Kemerer proposed six new object-oriented metrics to overcome the limitations of the more traditional code-based metrics. However, as computer code engineers‟ focus has shifted to the sooner stages of the life cycle, the shortcomings of OO code metrics like their predecessors became additional apparent. Therefore, a comprehensive approach to developing and applying metrics to artifacts like styles made at the first stages of the life cycle is required. within the in the meantime, the Unified Modeling Language (UML) was adopted by the thing Management cluster (OMG) in 1997 ending the questionable "OO strategies war", and since then has become the de facto specification customary graphical language for specifying, constructing, visualizing, and documenting computer code systems, business modeling and different non-software systems.
One of the earliest studies in this area was the development of software security design principles by Saltzer and Schroeder [10]. These principles were intended as guidance to help develop secure systems, mainly operating systems. Bishop‟s [11]. Many organizations square measure victimisation UML as normal|a typical} language for his or her project artifacts and have adopted UML as their organization's standard. because the quantity of UML models made inside a corporationinflated, a desire for measure their characteristics has arisen.

Bansiya, J., and Davis, C.G. proposed a set of metrics for object-oriented design called "Quality Model for Object-Oriented Design (QMOOD)" [10, 94]. This is a hierarchical

model for the assessment of high level design quality attributes of object-oriented design which is called as QMOOD. This model evaluates the structural and behavioral design properties of classes, objects and their relationships using a suite of objectoriented design metrics. This hierarchical model relates design properties of encapsulation, modularity, coupling, and cohesion to high-level quality attributes such as flexibility, reusability and complexity. The relationship from design properties to quality attributes are weighted in accordance with their influence and importance. A key attribute of the model is that it can be easily modified to include different relationships and weights. The model is empirically validated on large commercial object-oriented systems.

Aggarwal, K.K., et al. proposed a model for integrated complexity measurement for measuring the software complexity based on lines of code, average variable statement, cyclomatic complexity and degree of control nesting [3].

Chae, H.S., et al. presented an approach for improving the cohesion by considering the characteristics of the dependent instance variables in an object-oriented program. They investigated the effects of dependent instance variables on cohesion metrics for object-oriented programs and they proposed an approach to identifying the dependency relations among instance variables [18].

Aggarwal, K.K., et al. proposed two design metrics for object-oriented software and these metrics are analytically evaluated against Weyuker's properties of measures [4].

Aggarwal, K.K., et al. conducted an investigation on 22 metrics proposed by various researchers and applied these metrics on projects for empirical study [5].

Sarkar, S., et al. proposed a set of metrics that measure the quality of modularization of a non-object-oriented software system. They proposed design principles to capture the notion of modularity and they defined metrics centered on principles. Their metrics characterize the software from a variety of perspectives as structural, architectural, and notions such as the similarity of purpose and commonality of goals. Their metrics are based on information-theoretic principles and tested their metrics on popular open-source systems [79].

Aggarwal, K.K., et al. conducted effect of design metrics on fault proneness in object-oriented systems. They empirically investigated the relationship between object-oriented design metrics and fault-proneness of object-oriented systems [6].

Sarkar, S., et al. proposed 13 metrics for measuring the modularization of large-scale object-oriented software. Their 13 metrics characterise the quality of modularisation with respect to such object-oriented intermodule dependencies [78].

Alghamdi, J.S. presented a scheme for measuring coupling between program components. His scheme makes the measurement of coupling easier by breaking it down into two major steps and provides a systematic procedure for each step [8].

Kaur, K., and Singh, H. validated component based software development on reuse of software components. They have validated object-oriented metrics to measure structural properties of commercial software components [52].

Bawane, N., and Srikrishna, C.V. proposed a metric for software and the process of selecting the metrics that support the goal of measuring design and code quality [12].

Kaur, K., and Singh, H. conducted a study on system behavior for object-oriented systems using metrics. They conducted empirical studies using two object-oriented languages [53].

Object-oriented metrics can play an important role in object-oriented software development. The object-oriented metrics are important in the development of successful software applications [31].

Ma, Y.T., et al. (2010), proposed a hierarchical set of metrics for coupling and cohesion. They conducted empirical study on 12 open-source object-oriented software systems for validating their set. Their experimental results show the correlations between cross-level metrics and they provided more effective information about fault-prone classes in practice [66].

Kumar, S.A., et al. proposed the significance of software metrics to quantify design and code quality and discussed on the needs of development and implementation of metrics [62].

Okike, E. presented a pedagogic evaluation about the Chidamber Kemerer LCOM metric using field data from three industrial systems. They suggested that the LCOM metric measures class cohesiveness and appropriateness in the determination of properly and improperly designed classes [68].

Babu, S., and Parvathi, R.M.S. proposed an approach to the computation of dynamic coupling measures in distributed object-oriented systems. The motivation of measures is to complement existing measures that are based on static analysis by actually measuring coupling at runtime in the hope of obtaining better decision and prediction models [9].

Ahmed, M., and Shoaib, M. proposed design metrics to measure real time environment and the aim of the set of new metrics is to measure the design before handing over to the implementation team [7]. The measurement can distinguish the characteristics of entity from another by analysis and drawing the conclusion that software metrics are used to measure the attributes of an entity. It is accepted that quality of software product is strongly dependent on the quality of its design [83, 88]

Yadav, A., and Khan, R.A. proposed coupling metrics for complexity normalization. They proposed a method to improve reliability of object-oriented design by normalizing complexity which is closely correlated with coupling and coupling complexity normalization (CCN) metric is used to minimize complexity of object-oriented design [99].

Chhikara, A., and Chhillar, R.S. proposed an aspect-orientated object-oriented metrics. Aspect-Oriented Paradigm is the emerging paradigms that promise to enhance software design and promotes reuse. Their research studies the object-oriented metrics and how the introduction of aspects affects these metrics [22].

Chhikara, A., et al. conducted the impact of different types of inheritance on the object-oriented software. Their research paper focused on effects of inheritance on object-oriented environment [23].
Gandhi, P., and Bhatia, P.K. proposed two metrics called Message Received Coupling (MRC) and Degree of Coupling (DC) metrics for the automatic detection of design problems along with an algorithm to apply these metrics to redesign an object oriented source code. They designed a Method Calling Graph for calculating the value of proposed metrics [38].
Sharma, R., and Chhillar, R.S. discussed the merits and demerits of various metrics. They proposed a new system for measuring the goodness of implementation phase. The concept of object-oriented metrics has also been explored [84].
Sharma, A.H., et al. presented a review of the quality metrics suites of CK, MOOD, and LK metrics. They select some metrics and discard other metrics based on the definition and capability of the metrics [86].
Reda, S., et al. presented a methodology for software design quality assessment. Their methodology helps the designer to measure and assess the changes in design due to design enhancements. They illustrated the methodology using practical software design examples and analyzed its utility in industrial projects [75].
Kumar, R., and Gupta, D., proposed heuristics for object-oriented metrics. They proposed heuristics for CK, MOOD, and LK object-oriented metrics [60].
Krishnaiah, R.V., and Prasad, B.S. (2012), studied a suite of metrics for object-oriented design. The metric values have been calculated using a semi-automated tool. They analyzed the resulting values of CK and MOOD metrics and provided significant insight about the object oriented characteristics of the projects [59].
Dubey, S.K., and Rana, A. proposed a fuzzy model to quantify maintainability of object-oriented software system using Chidamber and Kemerer object-oriented metrics. The model takes object-oriented projects and evaluates its maintainability and fuzzy model is validated by using analytical hierarchy processing technique [32].
Dubey, S.K., et al. (2012), reviewed object-oriented metrics and they analyzed the difference between the object-oriented metrics and they studied object-oriented metrics which assures to reduce cost and the maintenance effort by serving as early predictors to estimate software faults [33].
Dash, Y, et al. (2012), studied artificial neural network and they explored the application of evaluate maintainability of the object-oriented software and they studied maintenance effort [30].
Chawla, M.K., and Chhabra, I. (2012), has conducted mapping of program characteristics into five structural complexity metrics and behavior of an information system. They applied and obtained results from three java based sorting programs [21].
Jyothi, V.E., et al. (2012), have studied agile software development refactoring to improve software quality and improve software internal structure without changing its behavior. They proposed an object-oriented software metric tool called “Metric Analyser” and the tool was tested on different codebases [47].
Gupta, A., et al. discussed the most commonly used metrics suite of CK, MOOD and LI on the basis of characteristic they measure. Further, they identified strengths and weaknesses of these metrics and concluded that none of the metrics suite is foolproof. Moreover, there is no single metric that can measure all the aspects of an object-oriented System [41].
Sharma, A.K., et al. reviewed the metrics of CK, MOOD, and LK metrics. They analyzed the metrics and recommended that are useful in evaluation of software quality [85].
Patidar, K., et al. (2013), presented a measurement of the coupling and cohesion between objects that measures the association between numbers of classes, check the direct dependencies, indirect dependencies, I/O dependencies, number of out and in metrics in object-oriented programming [71].
Michura, J., et al. proposed a set of metrics to quantify and measure the attributes. They proposed complexity metrics which are used to determine the difficulty in implementing changes through the measurement of method complexity, method diversity, and complexity density [67].
Lamrani, M., et al. (2013), presented an approach to express software design metrics based on a formal definition of the UML Meta model. They applied their approach to the well known suite of metrics called the CK metrics and MOOD metrics [63].
Chawla, S. (2013), has reviewed the set of MOOD and QMOOD metrics sets and they discussed the usefulness of each metrics [20].
Kaur, A., and Kaur, P.J. (2013), studied class cohesion metrics measured during the design phase to predict software quality. They used cohesion to evaluate class based on the information that is available during design phases [50].
In this paper we characterize several security outline measurements. They can be utilized to think about various plans for the same program and recognize the best outline for a certain security plan guideline. They do this by recognizing potential data stream taking into account examining the programming quality properties characterized in the QMOOD.

III. CONCLUSION

The design metrics play an important role in helping designer and developers to understand design aspects of software and it improves the software quality and productivity. In general, object-oriented metrics serve many purposes for software engineers and software metrics are used by the project manager, developer, and tester in assuring the quality of the software products. In today’s software development environment, object-oriented design and development is important and there is strong relationship between the object-oriented metrics and the testability efforts in object-oriented

system. This paper has analyzed the most referred object-oriented design metrics proposed by Chidamber and Kemerer, MOOD metrics set, and Lorenz and Kidd metrics. This paper also discussed the recently proposed "Comprehensive Metrics" suite for object-oriented design quality assessment and the review of object-oriented metrics proposed by various researchers and their significances are also outlined. The use of existing metrics and development of new metrics will be important factors in future software engineering process and product development [37, 46, 93, and 94]. In future, research work will be based on using software metrics in software development for the improvement of the time schedule, cost estimates and quality and can be improved through software metrics.

REFERENCES

[1]. Sachitano, R. O. Chapman, and J. A. Hamilton, "Security in software architecture: a case study," in Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004, pp. 370-376.

[2]. P. K. Manadhata, K. M. C. Tan, R. A. Maxion, and J. M. Wing, "An Approach to Measuring A System's Attack Surface," Carnegie Mellon University, Pittsburgh, PA August 2007.

[3]. I. Chowdhury, B. Chan, and M. Zulkernine, "Security metrics for source code structures," in Proceedings of the Fourth International Workshop on Software Engineering for Secure Systems Leipzig, Germany: ACM, 2008.

[4]. E. A. Schneider, "Security architecture-based system design," in Proceedings of the 1999 workshop on New Security Paradigms Caledon Hills, Ontario, Canada: ACM, 2000.

[5]. J. Bansiya and C. G. Davis, "A hierarchical model for object-oriented design quality assessment," IEEE Transactions on Software Engineering, vol. 28, pp. 4- 17, 2002.

[6]. J. Jürjens, Secure Systems Development with UML: Springer, 2005.

[7]. J. Barnes, High Integrity Software: The SPARK Approach to Safety and Security. London, Great Britain: Addison-Wesley, 2003.

[8]. G. McGraw, Software Security: Building Security In. Upper Saddle River, NJ: Addison-Wesley, 2006.

[9]. M. Howard and D. LeBlanc, Writing Secure Code.Redmond, Wash.: Microsoft Press, 2002.

[10]. D. Volpano, G. Smith, and C. Irvine, "A sound type system for secure flow analysis," Journal of Computer Security, vol. 4, pp. 167-187, January 1996.

[11]. A. Sabelfeld and A. C. Myers, "Language-based information-flow security," IEEE Journal on Selected Areas in Communications, vol. 21, pp. 5-19, 2003.

[12]. J. H. Saltzer and M. D. Schroeder, "The protection of information in operating systems," in Proceedings of the IEEE, 1975, pp. 1278-1308.

[13]. M. Bishop, Computer Security: Art and Science. Boston: Addison-Wesley, 2003.

[14]. Bandar Alshammari, Colin Fidge and Diane Corney," Security Metrics for Object-Oriented Class Designs", QSIC 2009 Proceedings of : Ninth International Conference on Quality Software , August 24-25, 2009, Jeju, Korea. (In Press)

[15]. K. Maruyama, "Secure refactoring: improving the security level of existing code," in Proceedings of the Second Internation Conference on Software and Data Technologies Barcelona, Spain, 2007.

[16]. J. Bansiya, "A Hierarchical Model for Quality Assessment of Object-Oriented Designs," Ph.D. Thesis, University of Alabama in Huntsville, 1997.

[17]. C. T. Wu, A comprehensive introduction to objectoriented programming with Java, 1st ed. Boston, MA: McGraw Hill Higher Education, 2008.

[18]. K. Maruyama and K. Tokoda, "Security-Aware Refactoring Alerting its Impact on Code Vulnerabilities," in 15th Asia-Pacific Software Engineering Conference, 2008. APSEC '08. , 2008, pp. 445-452.

[19]. L. C. Briand, J. W. Daly, and J. Wust, "A unified framework for cohesion measurement in object-oriented systems," in Proceedings of the Fourth International Software Metrics Symposium., 1997, pp. 43-53.

[20]. M. Fowler, Refactoring: Improving The Design Of Existing Code. Reading, MA: Addison-Wesley, 1999.

[21]. B. Chess and J. West, Secure programming with static analysis. Upper Saddle River, NJ: Addison-Wesley, 2007.

[22]. Abreu, F.B., and Melo, W., "Evaluating the Impact of Object-Oriented Design on Software Quality," Proceedings of the 3rd International Software Metrics Symposium, IEEE, Berlin, Germany, March, 1996.

[23]. Alemneh, E., "Current States of Aspect Oriented Programming Metrics," International Journal of Science and Research, Volume 3 Issue 1, January 2014, pp. 142-146.

[24]. Aggarwal, K.K., Singh, Y., and Chhabra, J.K., "A Multiple Parameter Software Complexity Measure," The Journal of Computer Society of India, Vol. 33, No 1, March 2003, pp. 22-30.

[25]. Aggarwal, K.K., Singh, Y., Kaur, A., and Malhotra, R., "Software Design Metrics for Object-Oriented Software," Journal of Object Technology, Vol. 6, No. 1, January-February 2006, pp. 121-138.

[26]. Aggarwal, K.K., Singh, Y., Kaur, A., and Malhotra, R., "Empirical Study of Object-Oriented Metrics," Journal of Object Technology, Vol. 5. No. 8, November-December 2006, pp. 149-173.

[27]. Aggarwal, K.K., Singh, Y., Kaur, A., and Malhotra, R., "Investigating effect of Design Metrics on Fault Proneness in ObjectOriented Systems", Journal of Object Technology, Vol. 6, No. 10, November-December 2007, pp. 127-141.

[28]. Ahmed, M., and Shoaib, M., "Novel Design Metrics to Measure Real Time Environment Application Design," Journal of American Science, 7(7), 2011, pp. 222-226.

[29]. Alghamdi, T.S., "Measuring Software Coupling," The Arabian Journal for Science and Engineering, Vol. 33, No. 1B, April 2008 , pp. 119-129.

[30]. Babu, S., and Parvathi, R.M.S., "Design Dynamic Coupling Measurement of Distributed Object Oriented Software Using Trace Events," Journal of Computer Science, 7 (5), 2011, pp. 770-778.
[31]. Bansiya, J., and Davis, C.G., "Hierarchical Model for Object-Oriented Design Quality Assessment," IEEE Transactions on Software Engineering, Vol. 28, No. 1, January 2002, pp. 4-17.
[32]. Basili, V.R., Briand, L.C., and Melo, W.L., "A Validation of Object-Oriented Design Metrics as Quality Indicators," IEEE Transactions on Software Engineering, Vol. 22, No. 10, October 1996, pp. 751-761.
[33]. Bawane, N., and Srikrishna, C.V., "A Survey of Quality Assessment through Object-Oriented Metrics," CSI Communications, Vol. 33, No 7, October 2009, pp. 21-25.
[34]. Bhatia, P.K., and Mann, R., "An Approach to Measure Software Reusability of OO Design," Proceedings of 2nd National Conference on Challenges and Opportunities in Information Technology, RIMT-IET, Mandi Gobindgarh, March 2008, pp. 26-30.
[35]. Bieman, J.M., and Ott, L, M., "Measuring Functional Cohesion," IEEE Transactions on Software Engineering, Vol.20, No.8, August 1994, pp. 644-657.
[36]. Briand, L.C., Morasca, S., Basili, V.R., "Property-Based Software Engineering Measurement," IEEE Transactions on Software Engineering, Vol. 22, No.1, January 1996, pp. 68-85.
[37]. Chae, H.S., and Bae, D.H., "Improving Cohesion Metrics for Classes by Considering Dependent Instance Variables," IEEE Transactions on Software Engineering, Vol. 30, No. 11, November 1998, pp. 826-832.
[38]. Chauhan, R., Singh, R., Saraswat, A., Joya, A.H., Gunjan, V.K., "Estimation of Software Quality using Object Oriented Design Metrics," International Journal of Innovative Research in Computer and Communication Engineering, Vol. 2, Issue 1, January 2014, pp. 2581-2586.
[39]. Chae, H.S., Kwon, Y.R., and Bae, D.H., "Improving Cohesion Metrics for Classes by Considering Dependent Instance Variables," IEEE Transactions on Software Engineering, Vol. 30, No. 11, October 2004, pp. 826-832.
[40]. Chandrika, S.M., Babu, E.S., and Srikanth, N., "Conceptual Cohesion of Classes in Object Oriented Systems," International Journal of Computer Science and Telecommunications, Volume 2, Issue 4, July 2011.
[41]. Chawla, S., "Review of MOOD and QMOOD Metric Sets," International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, Issue 3, March 2013, pp. 448-451.
[42]. Chawla, M.K., and Chhabra, I., "A Multiple Parameter Software Complexity Measure," International Journal of Engineering Research and Technology, Vol. 1 Issue 5, July 2012, pp. 1-5.
[43]. Chhikara, A., and Chhillar, R.S., "Impact of Aspect Orientation on Object Oriented Software Metrics," International Journal on Computer Science and Engineering, Vol. 2 No. 3, June-July 2011, pp. 466-469.
[44]. Chhikara, A., Chhillar, R.S., and Khatri, S., "Evaluating the Impact of Different Types of Inheritance on the Object Oriented Software Metrics," International Journal of Enterprise Computing and Business Systems, Vol.1 Issue 2 July 2011.
[45]. Chidamber, S.R., and Kemerer, C.F., "A Metrics Suite for Object-Oriented Design," IEEE Transactions on Software Engineering, Vol. 20, No. 6, June 1994, pp. 476-493.
[46]. Chidamber, S.R., Darcy, D.P., and Kemerer, C.F., "Managerial use of Metrics for Object-Oriented Software: An Exploratory Analysis," IEEE Transactions on Software Engineering, Vol.24, No. 8. August 1998, pp. 629 – 639.
[47]. Churcher, N.I., and Sheppard, M.J., "Comments on a Metrics Suite for Object – Oriented Design," IEEE Transactions on Software Engineering, Vol.21, No.3, March 1995, pp. 263-265.
[48]. Dallal, J.A., "Mathematical Validation of Object-Oriented Class Cohesion Metrics," International Journal of Computers, Issue 2, Vol. 4, 2010, pp. 45-52.
[49]. Dallal, J.A., "Measuring the Discriminative Power of Object-Oriented Class Cohesion Metrics," IEEE Transactions on Software Engineering, Vol. 37, No. 6, November-December 2011, pp. 788-804.
[50]. Dange, A.S., Joshi, S. D., " Fault Prediction in Object Oriented System Using the Coupling and Cohesion of Classes," International Journal of Computer Science and Management Studies, Vol. 11, Issue 02, August 2011, pp. 48-51.
[51]. Dash, Y, Dubey, S.A., and Rana, A., "Maintainability Prediction of Object Oriented Software System by Using Artificial Neural Network Approach," International Journal of Soft Computing and Engineering, Vol. 2, Issue. 2, May 2012, pp. 420-423.
[52]. Dubey, S.K., and Rana, A., "A Comprehensive Assessment of Object-Oriented Software Systems Using Metrics Approach," International Journal on Computer Science and Engineering, Vol. 2, No. 8, 2010, pp. 2726-2730.
[53]. Dubey, S.K., and Rana, A., "A Fuzzy Approach for Evaluation of Maintainability of Object Oriented Software System," International Journal of Computer Applications, Vol. 49, No 21, July 2012, pp. 1-6.
[54]. Dubey, S.K., Sharma, A., and Rana, A., "Comparison Study and Review on Object- Oriented Metrics," Global Journal of Computer Science and Technology, Vol. 12, Issue 7, April 2012, pp. 38-47.
[55]. Emam, K.E., and Melo, W., "The Prediction of Faulty Classes Using Object-Oriented Design Metrics," Institute for Information Technology, National Research Council, 1999, Canada.
[56]. Emam, K.E., Melo, W., and Machado, J.C., "The Prediction of Faulty Classes Using Object-Oriented Design Metrics," The Journal of Systems and Software, 56(2001), 2001, pp. 63-75.
[57]. Fenton, N.E., and Pfleeger, S.L., Software Metrics: A Rigorous and Practical Approach, Thomson Asia, Singapore, 2004.

[58]. Fernando, W.K.S.D., Wijayarathne, D.G.S.M., Fernando, J.S.D., Mendis, M.P.L., and Guruge, I., "The Importance of Software Metrics: Perspective of a Software Development Projects in Sri Lanka," Proceedings of the SAITM Research Symposium on Engineering Advancements, Sri Lanka, April 2014, pp. 91-95.

[59]. Gandhi, P., and Bhatia, P.K., "Optimization of Object-Oriented Design using Coupling Metrics," International Journal of Computer Applications, Vol. 27, No 10, August 2011, pp. 41-44.

[60]. Gelinas, J.F., Badri, M., and Badri, L., "A Cohesion Measure for Aspects," Journal of Object Technology, Vol. 5, No. 7, SeptemberOctober 2006, pp. 97 - 114.

[61]. Goyal, G., and Patel, S., "Analysis of Object Oriented Class Inheritance and Interfaces Using Coupling Measures," International Journal of Engineering and Social Science, Vol. 2, Issue5, 2012, pp. 93-103.

[62]. Gupta, A., Batra, G., and Vijaylaxmi., "Analyzing Theoretical Basis and Inconsistencies of Object Oriented Metrics," International Journal on Computer Science and Engineering, Vol. 4, No 5, May 2012, pp. 803- 808.

[63]. Harrison, R., Counsel, S.J. and Nithi, R.V., "An Evaluation of the MOOD Set of Object-Oriented Software Metrics," IEEE Transactions on Software Engineering, Vol.24, No.6, June 1998, pp.491-496.

[64]. Hitz, M., and Montazeri, B., "Chidamber and Kemmerer's Metrics Suite: A Measurement Theory Perspective," IEEE Transactions on Software Engineering, Vol.22, No4, April 1996, pp.267-271.

[65]. Jassim, F., and Altaani, F., "Statistical Approach for Predicting Factors of MOOD Method for Object Oriented," International Journal of Computer Science Issues, Vol. 10, Issue 1, No. 1, January 2013, pp. 589- 593.

[66]. Jayalakshmi, N and Satheesh, N., "Software Quality Assessment in Object Based Architecture," International Journal of Computer Science and Mobile Computing, Vol. 3, Issue. 3, March 2014, pg.941 – 946.

[67]. Jones, C., "Evaluating Software Metrics and Software Measurement Practices," Version 4, Namcook Analytics, March 2014. (http://Namcookanalytics.com).

[68]. Jyothi, V.E., Srikanth, K., and Rao, K.N., 2012, "Effective Implementation of Agile Practices – Object Oriented Metrics Tool to Improve Software Quality," International Journal of Software Engg. and Applications, Vol. 3, No 4, pp. 13-24.

[69]. Kan, S.H., Metrics and Models in Software Quality Engineering, Pearson Education, India, 2006.

[70]. Kanmani, S., Sankaranarayanan, V., and Thambidurai, P.,"External System Characteristics Assessment Using Object-Oriented Inheritance Metrics," The Journal of Computer Society of India, Vol. 32, No 2, June 2002, pp. 5-12.

[71]. Kaur, A., and Kaur, P.J.,"Class Cohesion Metrics in Object Oriented Systems," International Journal of Software and Web Sci., 3(2), Dec 2013, pp. 78-82.

[72]. Kapila, H., and Singh, S., "Bayesian Inference to Predict Smelly classes Probability in Open Source Software," International Journal of Current Engineering and Technology, Vol.4, No.3, June 2014, pp. 1724-1728.

[73]. Kaur, K., and Singh, H., "Metrics to Evaluate Object-Oriented Software Components" CSI Communications, Vol. 31, No 11, February 2008, pp. 23-26.

[74]. Kaur, K., and Singh, H., "Exploring Design Level Class Cohesion Metrics," Journal of Software Engineering and Applications, Vol. 3, 2010, pp. 384-390.

[75]. Kaur, K., and Singh, H., "An Investigation of Design Level Class Cohesion Metrics," International Arab Journal of Information Technology, Vol. 9, No. 1, 2012, pp. 66-73.

[76]. Kaur, M., Batra, P., and Khare, A., "Static Analysis and Run-Time Coupling Metrics," International Journal of Information Technology and Knowledge Management, Vol. 3, No. 2, July-December 2010, pp. 707-710.

[77]. Kaur, P.J., Verma, A., and Thapar, S., "Software Quality Metrics for Object-Oriented Environments," Proceedings of National Conference on Challenges and Opportunities in Information Technology, RIMT-IET, Mandi Gobindgarh, 2007, pp. 13-16.

[78]. Kitchenham, B., Pfleeger, S.L., and Fenton, N., "Towards a Framework for Software Measurement Validation," IEEE Transactions on Software Engineering, Vol. 21, No.12, December 1995, pp. 929-943.

[79]. Koh, T.W., Selmat, M.H., Ghani, A.A.A., and Abdullah, R., "Review of Complexity Metrics for Object Oriented Software Products," International Journal of Computer Science and Network Security, Vol.8, No.11, November 2008, pp. 314-320.

[80]. Krishnaiah, R.V., and Prasad, B.S., "Analysis of Object Oriented Metrics," International Journal of Computational Engineering Research, Vol. 2, Issue 5, September 2012, pp. 1474 – 1479.

[81]. Kumar, R., and Gupta, D., "Heuristics Based on Object Oriented (OO) Metrics," International Journal of Emerging Technology and Advanced Engineering, Vol. 2, Issue 5, May 2012, pp. 393-395.

[82]. Kumari, R and Jaspreet., "Implementation of Hybrid Metrics to Evaluate Software Performance Encapsulating CK Metrics," International Journal of Engineering Research and Technology, Vol. 2, Issue 3, March 2013, pp. 1-4.

[83]. Kumar, S.A., Kumar, T.A, and Swarnalatha, P., "Significance of Software Metrics to Quantify Design and Code Quality," International Journal of Computer Applications, Vol. 11, No.9, December 2010, pp. 36-42.

[84]. Lamrani, M., Amrani, Y.E., and Ettouhami, A., "A Formal Definition of Metrics for Object Oriented Design: MOOD Metrics," Journal of Theoretical and Applied Information Technology, Vol. 49, No 1, March 2013, pp. 1-10.

[85]. Lorenz, M., 1993, Object-Oriented Software Development, A practical guide, PTR prentice Hall, Englewood Cliffs, New Jersey, 1993.

[86]. Lorenz, M., and Kidd, J., Object-Oriented Software Metrics, Prentice Hall, Englewood Cliffs, New Jersey.
[87]. Ma, Y., He, K., Li, B., Liu, J., and Zhou, X., "A Hybrid Set of Complexity Metrics for Large-Scale Object-Oriented Software Systems," Journal of Computer Science and Technology, 25(6): November 2010, pp. 1184–1201.
[88]. Michura, J., Capretz, M.A.M., and Wang, S., "Extension of Object-Oriented Metrics Suite for Software Maintenance," ISRN Software Engineering, Vol. 2013, pp. 1-14.
[89]. Okike, E., "A Proposal for Normalized Lack of Cohesion in Method (LCOM) Metric Using Field Experiment," International Journal of Computer Science Issues, Vol. 7, Issue 4, No 5, July 2010, pp. 19-27.
[90]. Okike, E., "A Pedagogical Evaluation and Discussion about the Lack of Cohesion in Method (LCOM) Metric Using Field Experiment," International Journal of Computer Science Issues, Vol. 7, No 3, March 2010, pp. 36-43.
[91]. Olague, H.M., Etzkorn, L.H., Gholston, S., and Quattlebaum, S., "Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes," IEEE Transactions on Software Engineering, Vol. 33, No. 6, June 2007, pp. 402-419.
[92]. Patidar, K., Gupta, R., and Chandel, G.S., "Coupling and Cohesion Measures in OO Programming," International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, Issue 3, March 2013, pp. 517-521.
[93]. Pressman, R.S., Software Engineering a Practitioner's Approach, 5th Edition, McGraw Hill, India, 2001.
[94]. Pasupathy, S and Bhavani, R, "Analyzing the Efficiency of Program Through Various OOAD Metrics," Journal of Theoretical and Applied Information Technology, Vol. 61 No.2, March 2014. pp. 346-351.
[95]. Rajnish, K., "Another New Complexity Metric for Object-Oriented Design Measurement," International Journal of Hybrid Information Technology, Vol.7, No.2, 2014, pp.203-216.
[96]. Reda, S., Ammar, H., and Hegazy, O., "A Methodology for Software Design Quality Assessment of Design Enhancements," International Journal of Computer Science and Network, Vol. 1, Issue 6, December 2012, pp. 101-109.
[97]. Singh, H, Kumar, A, "A Novel Approach to Enhance the Maintainability of Object Oriented Software Engineering During Component Based Software Engineering," International Journal of Computer Sci. and Mobile Computing, Vol. 3, Issue 3, Mar 2014, pp.778 – 786.
[98]. Salem, A.M., and Qureshi, A.A., "Analysis of Inconsistencies in Object Oriented Metrics," Journal of Software Engg. and Applications, 4, 2001, pp. 123-128.
[99]. Sarkar, S., Kak, A.C., and Rama, G.M., "Metrics for Measuring the Quality of Modularization of Large-Scale Object-Oriented Software," IEEE Transactions on Software Engineering, Vol. 34, No. 5, September-October, 2008, pp. 700-720.
[100]. Sarkar, S., Rama, G.M., and Kak, A.C., "API-Based and Information-Theoretic Metrics for Measuring the Quality of Software Modularization," IEEE Transactions on Software Engineering, Vol. 33, No. 1, January 2007, pp. 14-32.