International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR)
ISSN 2249-6831
Vol. 3, Issue 2, Jun 2013, 283-294
© TJPRC Pvt. Ltd.

ESTIMATING RELIABILITY OF SOFTWARE SYSTEM USING OBJECT-ORIENTED METRICS

JOHNY ANTONY P[1] & HARSH DEV[2]

[1]Assistant Professor, Department of MCA, SHIMT, Sitapur, Uttar Pradesh, India

[2]Professor, Department of Computer Science, PSIT, Kanpur, Uttar Pradesh, India

ABSTRACT

Software quality has been a major challenge in the software industry. Reliability is one of most important quality factors of any software and an important issue in the development of successful applications. Software reliability is affected by many factors during the life cycle of a software product. Software metrics provide a quantitative basis for planning and measuring software development process. By the introduction of object oriented technology and programming, software designers experience another paradigm shift in the field of software engineering. Available traditional software metrics are not sufficient for evaluating various software products. As a result, a set of new object oriented software metrics came into existence. In this work, we use CK metrics for measuring the reliability of software. A tool is designed and developed called 'Java Class Analyzer' which extracts the values of the metric parameters from the source code. Based on the values of metric parameters extracted, some relationship is established between reliability and object oriented metrics. These results help the customer and producer to improve the software development process.

KEYWORDS: Object Oriented Metrics, Software, Reliability

INTRODUCTION

One of the biggest challenges of software industry is the difficulty of fulfilling the customers' precise demands. Large and complex systems have to be developed within in a schedule and budget, and after delivering them to the customer, they have to be maintained and improved based on the new requirements. Due to the complexity of problems and various other constraints, developers cannot always choose the best solution for each problem, so the quality of the software system may deteriorate over time. As Nina S Goodbole points out, "Quality goals must be clearly defined, effectively monitored and rigorously enforced" [19]. Quality must be defined and measured if excellence is to be achieved, business is to be successful. These thoughts bring to our mind that how we assess the quality of something intangible like software quality. To assess quality, the quality attributes must be taken into consideration and measured in the planning and design of the software [37]. At each and every stage, verification and validation are necessary factors for attaining quality. Verification refers: Are we building the product right? And Validation refers: Are we building the right product [19].

Reliability is one of the key factors which decide the quality of the system. Over 200 models have been developed since the early 1970, but now to quantify software reliability still remains largely unsolved. Several incidences have been reported because of the unreliable software. There is always increasing demand for reliable software. Software reliability has emerged as people try to understand the characteristics of how and why software fails [72]. Unlike other engineering disciplines, absolute measurements like mass, velocity are uncommon in software engineering. Instead, we attempt to derive a set of indirect measures that lead to metrics which provide an indication of the reliability of software. Metrics plays an important role in measuring the reliability of software. Metrics are the continuous application of measurement based techniques to the software development process and its products to supply meaningful information together with the use of techniques to improve the process and its products [32]. We need to identify necessary metrics that provide useful information to measure the reliability of software.

Overview of Software Reliability

The IEEE defines software reliability as “The ability of a system or component to perform its required functions under stated conditions for a specified period of time [27, 29, 31, 34]. The user oriented reliability of a program is defined as the probability that the program will give the correct output with a typical set of input data from the user environment [64]. Software reliability is the probability of failure free software operation which affects the system reliability and it differs from hardware reliability in that it reflects the design perfection rather than manufacturing perfection [53]. The high complexity of the software is a contributing factor towards the reliability problems. Good engineering methods can largely improve software reliability. Software reliability is a part of software quality. It relates to many areas where software quality is concerned. Hence measuring software reliability remains a difficult problem as we don’t have a good understanding of the nature of software. Reliability is measured as the probability that a system will not fail to perform its intended functions over a specified time interval. Customers are critically conscious of the reliability of software; they are likely to be largely unconcerned with the degree of the reusability of the components making up the source code. Amrit noted that “software reliability is a useful measure in planning and controlling resources during the development process so that high quality software can be developed” [2].

Objected oriented technology can provide a relationship between software development and software reliability. Object oriented software contains large number of attributes like inheritance, coupling, cohesion, class, polymorphism, which can provide more comprehensive description of software’s internal structure and nature. Therefore, it will be helpful to establish a relationship between object oriented metrics and reliability to know more about the internal structure and to motivate the designers to achieve the benefits and advantages of OOPS. Since reliability is an important quality factor of software system and among the large number of proposed object-oriented metrics, Chidamber and Kemerer (CK) [13] and MOOD metrics [1] demonstrated the strong impact on software quality and particularly on software reliability. This work analyzes the metric parameters of CK metrics to estimate the reliability of a software system.

LITERATURE REVIEW

Reliability Models

Software Reliability Models have emerged as people try to understand the characteristics of how and why software fails, and try to quantify software reliability. Over 200 models have been developed since the early 1970s, but how to quantify software reliability still remains largely unsolved. No single model completely represents software reliability. Among software reliability models, one can distinguish two main categories :- software reliability prediction model which address the reliability of the software early in the lifecycle and software reliability estimation model which evaluates current and future software reliability from failure data gathered beginning with the integration test of the software [32]. and McCall et.al. proposed a quality model which described reliability as accuracy, error tolerance, consistency and simplicity [28,47]. Boehm’s et.al. Presented reliability as self-containedness, accuracy, completeness, robustness and consistency. The source code possesses the characteristic reliability to the extent that it can be expected to perform its intended functions satisfactorily [9, 28]. FURPS model expressed reliability as that includes frequency and severity of failure, recoverability, predictability, accuracy, and mean time between failure [21, 36]. Dromey's model supported the view that reliability is the correctness of the software product [17]. According to ISO 9126 model, reliability is the extent to which a product is reliable. It is the result of factors like maturity, fault tolerance, recoverability and compliance. [35]. Lewis introduced Post Study System Usability Questionnaire (PSSUQ) which defined reliability as quantified consistency [43].

Barbacci et.al. described reliability as reliability as dependability where computer system can reasonably provide the service expected to and the attributes are availability, safety and security [5]. Rayleigh Model**,** among the family of Weibull distribution, is found to be most suitable for predicting reliability of software product. It predicts the expected value of defect density at different stages of life cycle of the project, once parameters (total number of defects or total cumulative defect rate and peak of the curve in terms of unit of time) for the curve are decided [30]. A few models like Jelinksi and Moranda De-Eutrophication model, Fault Count Model, Mills Seeding Model, Nelson Model Ramamurthy and Bastani Model repeatedly affirmed that reliability is fault free system and less failure count. All these models are based on the estimation of the number of initial faults present in a program component. Redundancy is often used to achieve high reliability [2]. Roger, in his User Oriented Software Reliability Model argued that reliable service can be provided to a user when it is know that error exists, provided that the service requested does not utilize the defective parts [62]. W. T. Tsai et.al. proposed a service-oriented software reliability model that dynamically evaluates the reliability of Web services [67].

According to Pabitra Kumar Panda Software Reliability is hard to achieve, because the complexity of software tends to be high. While the complexity of software is inversely related to software reliability, it is directly related to other important factors in software quality, especially functionality and capability. He assumes that reliability is a function of the defect level and as defects are removed, reliability improves [55].

Overview of Object-Oriented Metrics

Object-oriented design and development is becoming very popular in today's software development environment. Object oriented development requires not only a different approach to design and implementation, it requires a different approach to software metrics. Some of these metrics are very labor intensive to collect, or are dependent on the implementation environment. Metrics to measure software reliability exists and can be used starting in requirements phase to software design phase. The essential features of the OO metrics are the ability to cover all aspects of reliability because they address issues like class, inheritance, cohesion, coupling, methods, encapsulation etc. Number of object oriented metrics has been proposed in the literature [36] but only a few set of metrics are related with software reliability. CK metrics [13] can be used to analyze coupling, cohesion and complexity very well. Table 1 presents a list of object oriented metrics.

Table 1

Name	Source	Metrics
MOOSE/CK	Chidamber et.al. [13]	WMC, DIT, NOC, CBO, RFC, LCOM
MOOD	Abrreu et.al. [1]	MIF, AIF, MHF, AHF, POF, COF
LK	Lorenz et.al.[46]	CS, NOO, NOA, SI, OS, OC, NP
QMOOD	Bansiya [4]	DSC,NOH,NSI,NMI, NNC,NAC,NLC,ADI,AWI,ANA,MFM,
LiW	Li et.al. [44]	NAC, NLM,CMC,NDC,CTA,CTM
SATC	Rosenberg et.al. [63]	CC, LOC,WMC,RFC,LCOM,DIT,NOC
STREW-J	Nagappan et.al. [52]	NTC/SLC, NTC/NR, TLC/SLC,NA/SLC,NTC/NSC,NC,NLC/NC

Related Works

Numerous studies have empirically validated the association between OO metrics and quality of software. The selected literature includes OO metrics based prediction models and estimation models that focus on validating the effectiveness of OO metrics for either predicting or estimating fault-prone classes or quality of the system.

Sherry A.M., et.al. made study on object oriented software reliability models and proposed a new model stating the number of initial parameters serves as an important parameter of reliability model. He established a relationship between the number of initial faults present in an object and some metrics (CK) of OOPS [65]. Rosenberg Linda et.al. discussed how NASA projects in conjunction with SATC (Software Assurance Technology Centre) are applying software metrics to improve the quality and reliability of software products. Reliability is a by-product of quality and can be measured. Metrics used early can aid in detection and correction of requirement faults and guarantee reliability of the product [63]. Dolbec Jean and Shepard Terry proposed a component based software reliability model and it provides software system reliability estimates from the reliability of software components and the usage ratio of each component [16].

Hitz Martin and Montazeri Behzad measured product attributes of object-oriented system using object oriented metrics based on their effects on product attributes [26]. Chillar Usha and Bhasin Sucheta established a relationship between complexity of software and object oriented metrics. Complexity affects quality attributes like reliability, testability etc. [16]. Sharma Aman Kumar et.al identified a few object oriented metrics suitable for measuring the software quality and provided thresholds that could be used to judge the metrics collected from designs [64]. Nagappan Nachiappan et.al. developed a suite of metrics for early assessment of software reliability to provide feedback to the developer on the quality of their testing effort. The suite consists of easy method to analyze the information collected from the source code and test program [53]. Wildmaier C. James studies the strengths and weakness of the methodology of production through metrics in order to understand why they were not able to reach customer reliability objectives [71]. According to Quyoum Aasia, Software reliability is an important facet of software quality. It is dynamic and stochastic. He categorized reliability into modeling, measurement and improvement and then examined different modeling technique and metrics for software reliability [59]. Pandey Asheesh and Ahlawat Anil proposed application of neural networks for providing software reliability using object oriented metrics. He made use of complexity measures, cohesion and coupling measures as the independent variable. The validation has shown several well-known metrics can be profitably employed for the estimation of reliability [56]. Several other studies Helle [25], Varun Gupta [68], Dekkers [15], Klasky [42], Kaur [40], Raed [60], Arti [3], Khan [41], R.Subramnayam [65], Micheal [49], Yu [74], Gyimothy [22], and Deepak Arora [14] made use of object oriented metrics to make quality assessment of the software product. They provide useful feedback to the management to keep the software process and product more reliable.

RESEARCH OBJECTIVE

The main objective of this study is to analyze the relationship between CK Metrics with the reliability at the class level. Metric values are extracted from the projects to show how these metrics affects the reliability of the software product. CK metrics are defined to measure design complexity in relation to their impact on reliability. So we can relate the reliability and CK metrics to estimate the reliability of the product.

Research Hypothesis

The estimated value of reliability of the software is high with smaller values of WMC, RFC, LCOM, CBO, DIT, and NOC of the software.

MATHEMATICALLY

- Reliability is Inversely Proportional to WMC
- Reliability is Inversely Proportional to RFC
- Reliability is Inversely Proportional to DIT

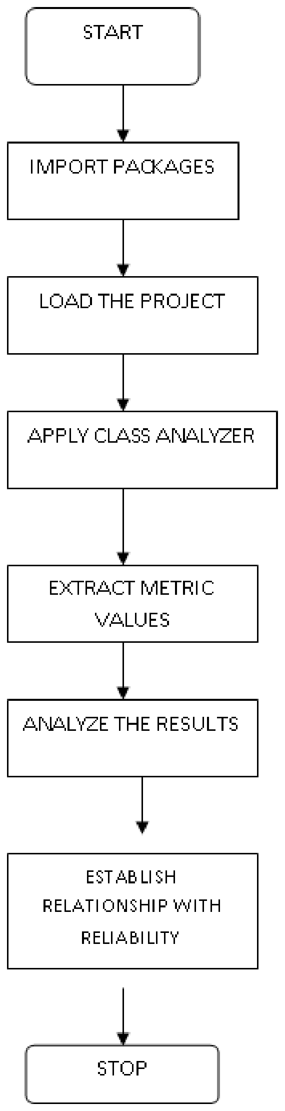
- Reliability is Inversely Proportional to LCOM
- Reliability is Inversely Proportional to CBO
- Reliability is Inversely Proportional to NOC

METHODOLOGY

The purpose of the study is to find out the relation between CK metrics and reliability of the software system. The systems that were analyzed are java applications. The CK metrics values are collected from the application using a specially developed Java Class Analyzer. For each class, ck metrics were collected.

Procedure to extract values of CK metric parameters from Java Projects

- Import necessary headers and packages
- Load the project
- Use the appropriate methods and procedures to retrieve the metric parameters from the project



Coding of the tool to extract WMC – (part of Java Class Analyzer)

```
import java.lang.reflect.*;
import java.io.*;
import java.lang.String.*;
import project1.myclass;
public class wmc{
public static void main(String args[]) {
try {
myclass d=new myclass();
Class c2=d.getClass();
System.out.println("Public Methods:");
Method methods[] = c.getDeclaredMethods();
System.out.println("Method Count ="+methods.length);
}
catch (Exception e) {
System.out.println("Exception: " + e);
}
}
}
```

DATA EXTRACTED FROM THE PROJECTS

Metrics	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Mean	SD
WMC	3	3	2	2	2	2	3	2	2	3	2.4	.45
RFC	2	3	4	2	4	2	3	2	3	3	2.7	.78
DIT	2	1	1	1	2	1	1	1	1	2	1.3	.45
LCOM	2	1	1	2	1	1	1	2	1	1	1.3	.45
CBO	1	1	2	1	2	1	2	1	1	1	1.2	.44
NOC	3	4	5	3	2	3	2	0	5	3	3	1.4

RESULTS AND DISCUSSIONS

The results obtained from the analysis of data supports the research hypothesis. The study proves that by keeping the low values of WMC, DIT, CBO, LCOM, RFC and high value of NOC, the designers can improve the reliability of the software and as a whole quality of the system.

Relationship between Reliability and WMC (Weighted Methods Per Class)

WMC is the count of number of methods in a class which defines the complexity of the class. As WMC increases, the complexity of the class also increases. According to CK, the larger the number of methods in a class the greater the potential impact on children. Children will inherit all the methods and it results in high complexity [13]. As a result the reliability of the system decreases. More number of methods is not recommended for achieving high reliability. Studies reported that a class with more member functions that its peers is considered to be more complex. The larger the WM, the larger the probability of fault detection and therefore more error prone [6, 22, 66]. Hence 3.1.1. is validated.

$$\text{Reliability } \alpha \ 1/\text{WMC} \quad \ldots\ldots\ldots\ldots\ldots \quad (1)$$

Relationship between Reliability and RFC (Response for Class)

RFC is defined as a count of the set of methods that can be potentially executed in response to a message received by an instance of the class. The test results show that classes with larger response sets requires more complex functionality and reduce the reliability of the system. According to CK, a class with large RFC indicates that a class can invoke larger number of methods. As a result the class becomes more complex and it is hard to maintain [13] and the larger the RFC, the larger the probability of fault detection [6, 22]. Hence 3.1.2. is validated.

$$\text{Reliability } \alpha \ 1/\text{RFC} \quad \ldots\ldots\ldots\ldots\ldots \quad (2)$$

Relationship between Reliability and DIT (Depth of Inheritance Tree)

DIT is the maximum inheritance path from the class to the root class. It assesses how deep a class is in hierarchy structure, the potential reuse of a class and its probable ease of maintenance. A project which is easy to maintain is highly reliable. A class with small DIT has much potential for reuse as it tends to be a general abstract class. On the other side, as a class gets deeper into a class hierarchy, it becomes more difficulty to maintain due to the increased complexity needed to capture its functionality. Basili et. al. empirically validated CK metrics and concluded that the larger the value of DIT, the greater the probability of fault-prone [6]. Deeper trees constitute greater design complexity, since more methods and classes are involved. As the class is deeper, the greater the number of method it is likely to inherit [13, 22]. A system with less depth of inheritance is easy to understand and to modify, therefore highly reliable. Hence 3.1.3. is validated.

Reliability **α** 1/DIT ………….. (3)

Relationship between Reliability and LCOM (Lack of Cohesion in Methods)

LCOM is the difference between the number of methods whose similarity is zero and not zero. The similarity of two methods is the numbers of attributes used were common. In object oriented software cohesion is often measured at class level. High cohesion and low coupling among classes are aimed at reduce complexity and increase reliability. A. Yadav and R.A. Khan observed that higher complexity makes the system more error prone, difficult to understand and unreliable [41, 72]. Low cohesion increases complexity, thereby increasing the likelihood of errors during the development process [13]. The smaller the LCOM value, the greater the reliability of a software. A high LCOM value generally pinpoints a poorly cohesive class. Hence 3.1.4. is validated.

Reliability **α** 1/LCOM ………….. (4)

Relationship between Reliability and CBO (Coupling Between Objects)

CBO for a class is count of number of other classes to which it is coupled. Strong coupling means that one object is strongly coupled with the implementation details of another object and it is discouraged because it results less flexibility, usability and reliability. Briand et.al. suggested that classes with more coupling and deep hierarchy are fault prone [10]. In order to improve encapsulation, inter-object class coupling should be kept minimum [13, 66]. Therefore lesser the coupling lesser will be complexity and greater will be the reliability of the system. Loose coupling increase reliability by decreasing complexity. The more the independent a class, it is easier to maintain and highly reliable. It is saying that coupling should be always minimum and cohesion should always be maximum. Hence 3.1.5. is validated.

Reliability **α** 1/CBO ………….. (5)

Relationship between Reliability and NOC (Number of Children)

NOC is a simple measure of the number of immediate sub classes associated with a given class using an inheritance relationship. It could be used to assess the potential influence that a class has on the overall design. It measures how many classes inherit directly methods from a super class This reduce the size of the super class or functions of the super class as the functions are divided among the subclasses. As Subrmanayam indicated that an increase in size of class is associated with a higher number of defects [66]. Researchers Basili et. al and Gyimothy et.al. observed that the larger the NOC, the lower the probability of fault detection [6, 22]. ie. classes with large NOC are less fault-prone. In our experiment, the value of NOC is found to be insignificant. Benlarbi et.al, in their analysis of thresholds values of CK Metrics, had not specified any threshold for NOC [7]. According to Yu et.al. NOC was significant and found that the more children a class has, the more fault prone it is which contradicts results of Basili. So We find difficult to state larger the NOC, the higher the reliability of the system. Therefore 3.1.6. is not validated, but we suggest that software with controlled value of NOC has good reliability. Therefore we add the value of NOC as an additional factor to estimate the reliability, as NOC alone can’t contribute to low/high reliability.

Reliability = Reliability estimated from other metrics +log (1+NOC) ………….. (6)

From (1), (2), (3), (4),(5), (6), We can say that Reliability = f(CK Metrics)

Reliability = (k *(1/(WMC*DIT*RFC*LCOM*CBO)) + log(1+NOC)

Value(WMC, DIT, RFC,LCOM,CBO) > 0 or ≤ threshold value

Where, the constant k will have to be worked out for a project based on the experience of the team members and characteristics of the software process.

Estimated Reliability with k = 1

Program	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
Reliability	0.64	0.81	0.84	0.72	0.5	0.85	0.53	0.125	0.76	0.65

CONCLUSIONS

This study made an assessment of the relationship between CK metrics and the reliability of objected oriented software system. We have selected all the CK metrics suite to estimate the reliability. The assessment proved empirically that by keeping WMC, DIT, CBO, LCOM, RFC and high value of NOC, the designers can attain high reliability of the system. Therefore we can say that CK metric parameters are useful indicators of the quality of the system. The size of the data set is small, the result is of limited capability. This paper did not validate other metric suite like MOOD, QMOOD etc. Validation of them will be of future work with larger data sets.

REFERENCES

1. Abreu, F.B: “The MOOD Metrics Set”, Proc. ECOOP’95, Workshop on Metrics.
2. Amrit, L. Goel (1985): “Software Reliability Models: Assumptions, Limitations, and Applicability”, IEET Transactions on Software Engineering, Vol SE-II, 12 December 1985 pages 1411-1423.
3. Arti, Chhikara., R. S. Chhilla and Sujata: “ Prediction Of The Quality Of Software Product Using Object Oriented Metrics”, International Journal Of Computer Engineering And Software Technology 2(1) Jan-June 2011; Pp. 1-6
4. Bansiya, J. and C. G. Davis : “A Hierarchical Model for Object-Oriented Design Quality Assessment”, IEEE Transactions on Software Engineering, 28, (1), (2002), 4–17.
5. Barbacci, Mario., Klein, Mark H., Longstaff, Thomas A., and Weinstock, Charles B: “Quality Attributes”, Technical Report of Software Engineering Institute Carnegie Mellon University Pittsburgh, Pennsylvania.
6. Basili, V.R., L.C. Briand and W.L. Melo: “A Validation of Object oriented design Metrics as Quality Indicators”, IEEE Transctions on software engneering, 22(10), 1996, pp 751-761.
7. Benlarbi, Saida., El-Emam, Khaled., Goel, Nishith and Rai N. Shesh: “Thresholds for Objected Oriented Measures”, http://dl.acm.org/citation.cfm?id=851024.856210.
8. Benlarbi, S K., EI Emamam., N. Goel and S. Rai: “Thresholds for Object Oriented Measures”, In 11th International Symposium on Software Reliability Engineering, 2000.
9. Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., McLeod, G., and Merritt, M: “Characteristics of Software

Quality", North Holland, 1978.

10. Briand, L., Daly, W. and Wust J, Unified Framework for Cohesion Measurement in Object Oriented Systems, Empirical Software Engineering, 65-117, 1998.
11. Chen, J and Y, Lum: "A New Metrics for Object-Oriented Design", Information of Software Technology 35, 4(April 1993):232-240.
12. Chhillar, Usha and Bhasin, Sucheta: "Establishing relationship between complexity and faults for objected oriented software system", IJCSI, Vol 8, Issue 5, No. 2 September 2011.
13. Chidamber, Shyam and Kemerer, Chris: "A Metric Suite for Object Oriented Design", IEEE Transactions of Software Engineering, June 1994.
14. Deepak, Arora., Pooja, Khanna., Alpika, Tripathi., Shipra, Sharma and Sanchika Shukla" " Software Quality Estimation through Object Oriented Design Metrics", IJCSNS International Journal 100 of Computer Science and Network Security, VOL.11 No.4, April 2011.
15. Dekkers, C.S: "The Secrets of Highly Successful Measurement Programs", Cutter IT Journal, vol 12 no. 4, pp. 29-35
16. Dolbec, Jean and Shepard, Terry: "A component based software reliability model" http://incoming-proxy.ist.edu.gr/stfs_public/cs/msc/Reading Material _MMSESEPE_oct2011/Software%20Engineering/Software%20Reliability%20Models/p19-dolbec.pdf
17. Dromey, R. G: "Concerning the Chimera [software quality]", IEEE Software, no. 1, pp. 33-43, 1996.
18. Fernando, Brito., Abreu and Walcelio, Melo: "Evaluating the impact of object oriented design on software quality", In METRICS '96: Proceedings of the 3rd International Symposium on Software Metrics, page 90, Washington, DC, USA, 1996. IEEE Computer Society.
19. Godbole, S. Nina: "Software Quality Assurance Principles and Practice" Narosa Publishing House.
20. Goodman, Paul: "Practical Implementation of Software Metrics" McGraw Hill, London.
21. Grady, R. B: "Practical software metrics for project management and process improvement", Prentice Hall, 1992.
22. Gyimothy, Tibor., Rudolf, Ferenc and Istvan, Siket: "Empirical validation of object oriented metrics on open source software for fault prediction", IEEE Trans. Softw.Eng., 31(10):897{910, 2005.
23. Harrison, R., S.J.Counsell and R.V.Nithi: "An evaluation of the MOOD set of OOSM", IEEE Transaction on Software Engineering, vol.24 no.6, pp.491-496, June 1998. JürgenWüst, "SD METRICS TOOL", in der Lache 17, 67308
24. Hector, M. Olague., Letha, H. Etzkorn., Sampson, Gholston and Stephen Quattlebaum: "Empirical validation of three software metrics suites to predict fault proneness of object-oriented classes developed using highly iterative or agile software development processes", IEEE Trans. Softw. Eng., 33(6):402{419, 2007.
25. Helle, Damborg Frederiksen**: "**Using Metrics in Managing and Improving Software Development Processes", Doctoral Consortium at ECIS 2001', June 2001, Department of Computer Science, Aalborg University, Denmark.
26. Hitz, Martin and Monazeri, Behzad: "Measuring Product Attributes of Object OrientedSystem",

http://citeseerx.ist.psu.edu/viewdoc/summary? doi=10.1.1.52.4579.

27. http://books.google.co.in/books?id=WSLzZc1ANqEC&pg=PA5&lpg=PA5&dq=definitions+of+software+reliability&source=bl&ots=YTwcy22wY3&sig=CySnHdqYCqtJK6XEMVTrQUL93po&hl=en&sa=X&ei=n91wUYq8JYiIrQfy6YCoBg&ved=0CEsQ6AEwAw#v=onepage&q=definitions%20of%20software%20reliability&f=false
28. http://www.bth.se/com/besq.nsf/(WebFiles)/CF1C3230DB425EDCC125706900317C44/$FILE/chapter_1.pdf.
29. http://www.businessdictionary.com/definition/software-reliability.html
30. http://www.tataelxsi.com/whitepapers/SoftReliabilityPredictModel.pdf
31. https://ece.uwaterloo.ca/snaik~/ch15-softwarerelaibility.ppt
32. http://www.cs.ucl.ac.uk/staff/ucacwxe/lectures/3C05-01-02/aswe12.pdf
33. Hu, Q.P., Dai Y.S., Xie, M and Ng S.H: "Early Software Reliability Prediction with Extended ANN Model", Proceeding of the 30th Annual International Computer Conference COMPSAC, 2006
34. IEEE Standard (1991) 610.12-1990 Glossary of Software Engineering Terminology.
35. ISO, International Organization for Standardization, "ISO 9126-1:2001, Software engineering – Product quality, Part 1: Quality model", 2001.
36. Jacobson, I., Booch, G and Rumbaugh, J: "The Unified Software Development Process*",* Addison Wesley Longman, Inc., 1999
37. Jamali, Seyyed Mohsen:, "Object oriented metrics":(A survey approach) ttp://www.cs.sfu.ca/~sja25/personal/resources/papers/ObjectOrientedMetrics.pdf
38. Kan, H. Stephen: "Metrics and Models in Software Quality Engineering", Prentice Hall, 2003
39. Kaner, Cem: "Software Engineering Metrics: What do they Measure and How Do We Know", 10 th International Software Metrics Symposium*, 2004.*
40. Kaur, Kuljit: "Process and Product Metrics to Assess Quality of Software Evolution", Conference Proceedings, International Conference on Recent Advances and Future Trends in Information Technology (iRAFIT), 2012
41. Khan, R.A., K. Mustafa and S. I. Ahson**: "**An Empirical Validation of Object Oriented Design Quality Metrics", J. King Saud Univ., Vol. 19, Comp. & Info. *Sci*., pp. 1-16, Riyadh (1427H./2007).
42. Klasky, B. Hilda: "A Study of Software Metrics", A thesis submitted to the Graduate School-New Brunswick Rutgers, The State University of New Jersey, 2003.
43. Lewis, J.R: "IBM Computers usability satisfaction questionnaires: Psychometric evaluation and instruction for use", International Journal of Human computer interaction, 1995. pp 155-162.
44. Li, W and Sallie, Henry: "Metrics for Object-Oriented system", Transactions on Software Engineering, 1995
45. Lionel, C. Briand and JÄurgen, WÄust: "Empirical Studies of Quality Models in Object-Oriented Systems", In Advances in Computers, volume 56, September 2002.
46. Lorenz, M and J. Kidd: "Object Oriented Software Metrics", Prentice Hall, NJ, (1994).
47. McCall, J. A., Richards, P. K and Walters, G. F: "Factors in Software Quality", National l Technical Information

Service, Springfield, VA, 1977

48. Mei-Huei, Tang, Ming-Hung Kao, and Mei-Hwa Chen: “An empirical study on object-oriented metrics”, In METRICS '99: Proceedings of the 6th International Symposium on Software Metrics, page 242, Washington, DC, USA, 1999. IEEE Computer Society.
49. Michael, English., Chris, Exton., Irene, Rigon, and Brendan Cleary” “Fault detection and prediction in an open-source software project”, In PROMISE '09: Proceedings of the 5th International Conference on Predictor Models in Software Engineering, pages 1{11, New York, NY, USA, 2009. ACM.
50. Morris, K: “Metrics for Object-oriented Software Development Environments,” Masters Thesis, MIT, 1989.
51. Nagappan N., T. Ball and A. Zellar: “Mining Metrics to Predict Component Failures”, ICSE Proceedings of the 28th international conference on software engineering, Pages 452 – 461, New York, 2006, ACM Press
52. Nagappan, Nachiappan., Williams, Laurie, and Vouk, Mladen: “Towards a metric suite for early software reliability assessment”. http://research.microsoft.com/en-us/people/nachin/publications.aspx.
53. Nagappan, Nachiappan., Williams, Laurie., Vouk, Mladen and Osborne, Jason: “Usinf In-Process testing metrics to estimate software reliability: A Feasibility Study”, http://collaboration.csc.ncsu.edu/laurie/Papers/ISSRE_STREW-J.pdf.
54. Pan, Jiantao: “ Software Reliability”, http://www.ece.cmu.edu/~koopman/des_s99/sw_reliability/
55. Panda Pavitra Kumar: “Software Reliability”, 2010, http://www.slideshare.net/AnandKumar87/software-reliability-11841804
56. Pandey, Asheesh and Ahlawat, Anil: “Reliability and Maintainability of Software using object oriented metrics by artificial neural network”, IJMET, Vol.1, Issue 1, March 2013.
57. Parvinder, Singh Sandhu and Dr. Hardeep, Singh: “ A Critical Suggestive Evaluation of CK Metric” (www.pacis-net.org/file/2005/158).
58. Ping, Yu., Tarja Systa and Hausi, Muller: “Predicting Fault-Proneness using OO Metrics: An Industrial Case Study”, In Sixth European Conference on Software Maintenance and Reengineering *(CSMR 2002)*, pages 99{107, March 2002.
59. Quyoum, Aasia., Din, Dar and Mehar, Ud Quadri S.M.K., “Improving software reliability using software engineering approach – A Review”, International Journal of Computer Application, Vol. 10, No. 5, November 2010.
60. Raded Shatnawi, “An Investigation of CK Metrics Thresholds”. ISSRE 2006 Supplementary conference proceedings (www.computer.org/portal/web)
61. Roger C. Chenung, A user oriented software reliability model, IEET Transactions on Software Engineering, Vol SE-6, March 1980 pages 118-129.
62. Rosenberg Linda, Hammer Ted, and Shaw Jac, Software metrics and reliability http://www.google.co.in/#hl=en&rlz=1R2ASUM_enIN517&sclient=psy-ab&q=software+metrics+and+reliability+linda+rosenberg&rlz=1R2ASUM_enIN517&oq=sofware+metrics+and+reliability+lind&gs_l=hp.1.0.0i22i30.1716.13541.0.15429.50.38.6.0.0.2.982.13805.2-

1j24j1j6j2.34.0...0.0...1c.1.9.psy-ab.VtRQal9pyYY&pbx=1&bav=on.2,or.r_qf.&bvm=bv.45512109,d.bmk&fp=3833cd2675a87390&biw=1440&bih=600

63. Sharma, Aman Kumar., Kalia,Arvind and Singh, Hardeep: “Metrics identification for Measuring object oriented quality”, IJSCE, Vol. 2, Issue- 5, November 2012.

64. Sherry,A.M., A.K. Malviya and R.C. Tripathi: “A study of object oriented software reliability models” http://www.bvicam.ac.in/news/ INDIACom%202009%20 Proceedings/pdfs/papers/INDIACom09_262_Paper.pdf

65. Subramnayam R. and M.S. Krishnan: “Empirical Analysis of CK Metrics for Object Oriented Design Complexity: Implications for software Defects”, IEEE Trans. Software Eng, 2003

66. Tsai, W. T. , D. Zhang, Y. Chen, H. Huang, R. Paul*, N. Liao, “A SOFTWARE RELIABILITY MODEL FOR WEB SERVICES” http://webdocs.cs.ualberta.ca/~sr16/Reliability/A%20SOFTWARE%20RELIABILITY%20MODEL%20FOR%20WEB%20SERVICES.pdf

67. Varun Gupta, Object Oriented Static and Dynamic Software Metrics for Design and Complexity, Ph.D Thesis submitted to Department of Computer Engineering and NIT, Kusukshetra, 2010.

68. Victor R. Basili, Lionel C. Briand, and Walcelio L. Melo. A Validation of Object-Oriented Design Metrics as Quality Indicators. In *IEEE Transactions on Software Engineering*, volume 22, pages 751{761, October 1996.

69. Wesner J.M. et al, (1995), “ Winning with Quality”**,** Addison-Wesley, Reading, MA.

70. Wildmaier, C James: “Producing more reliable software: Mature Software Engineering process Vs. State-of-the-art Technology?” http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=870400&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D870400.

71. Yadav A, and R.A. Khan, “Impact of Cohesion on Reliability”. Journal of Information and Operations Management” Vol 3, Issue 1, 2012, PP 191-193.

72. Yadava Amitabha and Khan R.A., (2012), “Reliability Estimation Framework – Complexity Perspective” retrieved from http://airccj.org/CSCP/vol2/csit2509.pdf

73. Yu Ping, Tarja Syst, and Hausi A. Muller. Predicting fault-proneness using oo metrics: An industrial case study. In CSMR '02: Proceedings of the 6th European Conference on Software Maintenance and Reengineering, pages 99 107, Washington, DC, USA, 2002. IEEE Computer Society.

74. Yuming Zhou and Hareton Leung. Empirical analysis of object-oriented design metrics for predicting high and low severity faults. IEEE Trans. Softw. Eng.,32(10):771-789, 2006