JOURNAL OF SOFTWARE ENGINEERING & INTELLIGENT SYSTEMS
ISSN 2518-8739
30th April 2017, Volume 2, Issue 1, JSEIS, CADMEI Copyright © 2016-2017
www.jseis.org

A survey of software quality metrics for software measurement process

[1]Anam Zai, [2]Rawish Butt, [3]Shiza Nawaz
[1,2,3]International Islamic University, Islamabad, Pakistan
Email: [1]anam.zai@outlook.com, [2]rawish.butt@outlook.com, [3]shizaawan90@gmail.com

ABSTRACT

Software measurement process is a process that measures, adjusts, evaluates, and improves the software development process. Software metrics may be used in different models to improve software quality. This paper aims to identify software metrics to increase knowledge of the reasons and effects of using metrics and to evaluate their applicability in software development. Therefore, we use previous studies in the literature in order to describe the fundamental aspects of the software quality metrics. We discuss several metrics of software quality metrics: product quality, in-process quality, testing quality, and customer satisfaction quality. Process metrics have been seemed more successful in discovering faults as compared to complexity metrics and traditional size.

Keywords: software measurement process, software quality, software metric; survey;

1. INTRODUCTION

Quality is performance to the standard expected by the customer and applies to products, services, people, and processes. Quality must give superior value to the customer (superior value has 3 elements: superior cost, superior quality and superior services) [1]. Software quality assurance is one of the most important phases of software project management [2]. To ensure quality, software measurement has become a key aspect of good software engineering practice [3]. Software measurement process must be a good oriented systematic process that measures, calculates, corrects, and improves the software development process [4]. Measurement activates inform and involve software developer in all phases of software development process [3]. Software measurement increases the effectiveness of testing process [5]. A metric is a countable measurement of software product, project, or process that is directly calculated, observed and predicted [6]. Metrics provide visibility and insight about what we do and how well we did it [7]. Moreover, metrics should be able to help in developing models, which are useful in calculating process of the product spectrum [8]. Metrics can be used for process improvement and monitoring, product development, software estimations and for quality control [9]. There are some metrics quality characteristics defined by Latva-Koivisto [10] for good complexity metrics such as validity reliability etc.

Software metrics are valuable object in the whole software life cycle [11]. Software metrics contracts with the measurement of software product development process and help in evaluating models and tools [12]. Software metrics are used to assess the quality of the resulting product and the software development process [3]. Using software metrics we can expand variety of information about the quality of the product that is provided to the customer, cost estimation, progress of a software project, and complexity of a software system [13]. Software metrics used in the traditional software development process can be categorized into three types as product, process, and project metrics [14, 15]. Process metrics highlights the process of software development and used to augment software development and maintenance [16]. Project metrics are used to control project status and situation [17]. Product metrics define the attributes of the software product at any level of its development. Software product quality consists of two levels: intrinsic product quality and customer satisfaction. Intrinsic product quality is commonly measured by the number of "bugs" that can be functional defects in the software or by how long the software can execute before facing a "crash." [18].

2. RESEARCH BACKGROUND

Software quality is defined as a characteristic, an attribute, property, that parallels to the level of excellence, superior excellence and high input granularity [19]. The development of software applications contains concepts such as planned level, estimated level, and actual level of quality [20]. The planned quality level must be set in order to ensure getting the objective for which the application is being established, to achieve a level of effectiveness as high as possible [21]. The estimated quality level is used when the software application is not still offered or is in the process of elaborating the project's financial offer required within the program in order to define the necessary

resources and financial objectives [22]. The actual quality level of a software product is the outcome of measures executed within the exploitation period and is achieved by storing data into a model [11, 23]. Software Measurement is a important practice to process quality improvement and project management [24, 25]. SM is also a key discipline in assessing the quality of software products and the ability and performance of software processes. The software measurement process includes: measurement planning, measurement execution, and measurement evaluation [26]. After plan measurement can start. Measurement execution involves in gathering data for the defined measures, analyzing and storing them. The data analysis delivers information to decision making, supporting the identification of proper actions. Finally, the measurement process and its products should be estimated in order to categorize potential improvements [27]. Question arises what to measure and this question has two levels: what quality attributes to measure? What parts of the system should be measured? To measure quality select the correct measure from the huge collection of available software quality metrics [28].

Software metric is a field of software engineering that is associated with various amounts of computer software and its developments [29]. Software metrics is one of the important tools for analyzing the software product in an effective way [30] [31]. Software metric are helpful in enhancing the quality of software, cost estimation and planning the budget [32] and with the help of software metric we are capable to understand the software product in an effective way [33]. Metrics that help to measure the success or failure of a project are very different and these metrics hardly have a good contract in commonalities. Software quality metrics are the subset of software metrics that emphases on the quality aspect of software therefore play an important role in analyzing and improving software quality [34]. Software metrics values are the signs of one or more software quality attributes [35]. Software Quality Factors (SQFs) are attributes that seems important for software to possess [36]. There are many Quality Factors such as clarity, completeness, complexity, consistency, correctness, efficiency, flexibility act, and the significance of each depends on a given measure and definition. A quality factor can be used at any time in the lifecycle of the product. Metrics are useful to define current status of a project and estimate its performance. Many software measures events have been offered in the literature, some of them are [12, 37, 38] [39, 40]. Software metrics can be classified into three categories: project metrics, product metrics and process metrics [31].

3. PROJECT METRICS

Project metrics are used to control project situation and status. Project metrics prevent the problems or potential risks by standardizing the project and help to improve the software development plan [41]. Some authors consider that management of projects is management risks [42]. This fact is due, in part, by the understanding that a significant portion of projects failures could be related to a poor risk management [43]. Project metrics describe the project features and implementation. Examples include the number of software developers, the staffing outline over the life cycle of the software, schedule, productivity and cost [41].

4. PRODUCT METRICS

Product metrics define the attributes of the software product at any stage of its development. Product metrics may measure the size of the program, complexity of the software design, portability, maintainability, product scale and performance. Product metrics are used to suppose and discover the quality of the product. Product metrics are used to measure the standard or the final product [8, 44]. Product quality is a critical competitive issue when introducing new products [45]. Software product quality consists of two levels: intrinsic product quality (reliability, defect density) and customer satisfaction (customer problems, customer satisfaction). Intrinsic product quality is measured by the number of functional defects in the software or by how long the software can run before facing a failure. Reliability is the probability of a program will perform its specified function or a stated time period under specified condition [46]. Software reliability is important because it highly impacts reputation of a company, maintenance cost (warranty), future business, contract requirements and customer satisfaction. There are three metrics in software reliability Mean time between failures (MTBF), mean time to failure (MTTF), and mean time to repair (MTTR) [47]. MTTR is only applicable for repairable systems. Mean time between failures is predicted time between two succeeding failure of a system and that is stated in hours. It's a key reliability metric for system that can be fixed or restored and applicable when several systems failures are expected. Mean time to failure is the expected time to failure of a system [48]. MTTR metric is used with safety critical systems such as the avionics, airline traffic control systems and weapons. MTTF metric is more difficult to implement and may not be representative of all customers. MTTF is applicable for non-repairable systems. Mean time to repair is an average time to restore a system after failure [49].

The defect density metric, in contrast, is used in many commercial software systems. The overall idea of defect rate is the amount of defects over the opportunities for error (OFE) during a precise time frame. In time frames, several operational definitions are used for the life of product (LOP), extending from one year to many years after the software product's release to the overall market. Normally more than 95% of the bugs are found within four years of the software's deployment [11] [4]. For application software, most bugs are normally found within two years of its release. A line of code could be any line of program text that is not a blank line or comment, nevertheless the fragments of statements or the number of statements on the line [50]. The line of code (LOC) metric is simple but the main problem arises from the ambiguity of the operational definition. Differences between instruction, statements physical lines (or logical lines of code) and differences among languages commits to the huge differences in counting LOCs. Even within the same language, the approaches and algorithms used by various counting tools can cause important differences in the final counts [51]. Jones (1986) describes several variations: Count only executable lines, Count executable lines, data definitions, and comments, Count executable lines plus data definitions, Count lines as physical lines on an input screen, Count executable lines, data definitions, comments, and job control language, Count lines as terminated by logical delimiters [11]. To calculate defect rate for the modified code, LOC count and Defect Tracking must be presented. LOC count; the entire software product must be available. Defect tracking: Defects must be tracked to the release source, a lot of the code that covers the defects and when the portion was added, enhanced or changed [14].

The two metrics (Reliability and Defect Rate) are correlated but are different enough to merit close attention as both are predicted values and estimated using static and dynamic models. First, one measures the time between failures, the other measures the defects comparative to the software size (Function points, lines of code, etc.). Secondly it is difficult to distinct failures and defects in actual measurements, failures data tracking, and defects (or faults) have different meanings [11] [52].

Another product quality metric used by major developers in the software industry measures the problems customers encounter when using the product is Customer Oriented metrics [53]. Customer Oriented metrics involve Customer Problems Metric and customer satisfaction Metrics. Customer problems metric; problems that are not accurate defects may be unclear documentation or information, usability problems, duplication of user error or valid defects [54]. The problems metric is usually expressed in terms of problems per user month (PUM): PUM is equal to the total problems that customers reported (true defects and non-defect-oriented problems) for a time period per Total number of license-months of the software during the period [4] [11]. Customer satisfaction can be measured by customer survey data through the five-point scale: very satisfied, satisfied, neutral, dissatisfied and very dissatisfied. Satisfaction with the quality of the product and its measurements is typically found through various methods of customer surveys [55] [11].

5. PROCESS METRICS

Process metrics focuses on the process of software development. It mainly targets at process duration, type of methodology used and the cost acquired. Process metrics can be used to enhance software development and maintenance [56]. Effectiveness of defect removal during development, the modeling of testing defect onset, and the reaction time of the fix processes are the examples of process metrics [8]. Researchers have generally focused on two broad categories of metrics for fault prediction: code metrics, which measure properties of the code such as size and complexity and process metrics, are number of changes and number of developers [57]. Fault prediction models are used to enhance software quality and to support software inspection by locating possible faults [58]. Model performance is influenced by a modeling technique [59] [60] [61] [62]and metrics [63] [64] [65] [66]. Contradictory results through studies have often been stated. Even inside a single study, different results have been found when different environments or approaches have been used. But, finding the suitable set of metrics for a fault prediction model is still essential, because of major challenges in metrics performance [67].

Software metrics can be divided into different categories while same metrics may relate to more than one category. Some notable metrics that are classified into five categories commercial perspective, significance perspective, observation perspective, measurement perspective and software development perspective [3].

a) Commercial perspective

From Commercial Perspective, Metrics can be categorized into five classes (Technical Metrics, Defect Metrics, End-User Satisfaction Metrics, Warranty Metrics and Reputation Metrics) to measure the quality and quantity of software [3].

b) Significance perspective

From significance perspective, Metrics can be clustered into two classes: Core Metric; essential to maintain solution delivery test management on software systems development projects and Non-Core Metric (optional metric) help to produce a more stable picture of the quality and efficiency of test efforts [3].

c) Observation perspective

In observation perspective, Metrics can also be categorized as: Primitive metrics are directly detected, such as the program size (in LOC), total development time for the project and number of defects perceived in unit testing. Computed metrics cannot be directly detected but are added in some way from other metrics. Direct measurement is assessment of something existing and involved in measurement perspective [6]. Software metrics can also be classified as: Objective metric, same values for a given metric and Subjective metrics may measure changed values for a given metric, since their subjective result is involved in arriving at the measured value [12].

Since the 1970s, various common software quality models came out, such as McCall [6] [68] [31], Boehm [11] , FURPS [69] [70], ISO/IEC9126:1991 and ISO/IEC9126:2001 [71] [72]. Software Quality Model is generally defined as a set of characteristics and relations between them which actually offer the source for evaluating quality and specifying requirements of quality and comparing of SWQMs [68]. In 1977 jam McCall presented Quality model [73]which is known as Mc Call Quality Model and is mainly aimed to the system developers and the system development process. McCall model offered the relationship between software quality characteristics and software metrics, without software functionality. McCall quality model try to bridge the gap between users and developers concentrating on a number of software quality factors that reveal users' views and the developers' priorities [74]. The McCall quality model has three major perspectives Product revision, Product transition and Product operation for defining and identifying the quality of a software product [73, 75]. Product revision: Includes maintainability (find and repair a fault in the program within its operating environment), flexibility (the ease of change required by operating environment) and testability (the ease of testing the program, to confirm that it is error-free and fulfil its specification) [76] [77]. Product transition: It is all about portability (the struggle required to transfer a program from one environment to another), reusability (the ease of reusing software in a different situation) and interoperability (the effort required to combine the system to another system) [78] [76]. Product operation: The Quality of product processes depends on correctness (the degree to which a program accomplishes its specification), reliability (the system's capability not to fail), efficiency (use of resources, e.g. processor storage, time), integrity (the security of the program from unauthorized access) and usability (the ease of the software use) [79] [76] . Precise measurement is essential for all engineering disciplines, and Software engineering is not an exception. Engineers and researchers are seeking to express features of software in order to simplify software quality assessment. Large bodies of software quality metrics have been established, and number of tools exists to collect metrics from program representations. This variety of tools allows a user to select the tool best suited, depending on its handling, price or tool support. However, it is considered that all metrics tools compute, implement and interpret the same metrics in the same way [11] [80].

6. CONCLUSION

This paper is an introduction of software quality measure and metrics found in the software quality literature. Software measurement and metrics helps to evaluate software process, software project and software product. The set of metrics identified in this paper provide the better validation activity to an organization and develop better software process towards the goal of process management. Process metrics have been reported more effective in finding faults compared to complexity metrics and traditional size. The metrics' effectiveness is measured to provide a clear distinction about their usefulness. The effectiveness of the frequently used metrics were measured to distinguish between metrics usefulness and to define the degree of effectiveness. Well-designed metrics with documented objectives can help any organization to find the information it needs to improve its software processes, product and customer services. Therefore, future research is needed to improve the methodology to encompass metrics that have been validated on project and valid measures of quality on software project.

ACKNOWLEDGEMENT

We would like to thank International Islamic University, Islamabad Pakistan for providing support in order to complete this research. Moreover, we are also thankful to all individuals who provided insight and expertise that greatly assisted the research.

REFERENCES

1. Padma, P. and U. Wagenseil. *Antecedents of Service Excellence*. in *4 th International Conference on Contemporary Marketing Issues ICCMI June 22-24, 2016 Heraklion, Greece*. 2016.
2. Xu, J., D. Ho, and L.F. Capretz, *An empirical study on the procedure to derive software quality estimation models.* arXiv preprint arXiv:1507.06925, 2015.
3. Farooq, S.U., S. Quadri, and N. Ahmad, *Software measurements and metrics: Role in effective software testing.* International Journal of Engineering Science and Technology (IJEST), 2011. **3**(1): p. 671-680.
4. Ming-Chang, L., *Software Quality Factors and Software Quality Metrics to Enhance Software Quality Assurance.* British Journal of Applied Science & Technology, 2014. **4**(21): p. 3069.
5. Quadri, S. and S.U. Farooq, *Software testing–goals, principles, and limitations.* International Journal of Computer Applications, 2010. **6**(9): p. 7-10.
6. Futrell, R.T., L.I. Shafer, and D.F. Shafer, *Quality software project management*. 2001: Prentice Hall PTR.
7. Pfleeger, S.L., *Software metrics: progress after 25 years?* IEEE Software, 2008. **25**(6): p. 32.
8. Rawat, M.S., A. Mittal, and S.K. Dubey, *Survey on impact of software metrics on software quality.* IJACSA) International Journal of Advanced Computer Science and Applications, 2012. **3**(1).
9. Siakas, K.V., E. Georgiadou, and E. Berki. *Agile methodologies and software process improvement*. in *IADIS (International Association for development of the Information Society) International Virtual Multi Conference on Computer Science and Information Systems (MCCSIS 2005)-SEA (Software Engineering and Applications)*. 2005.
10. Polančič, G. and B. Cegnar, *Complexity metrics for process models – A systematic literature review.* Computer Standards & Interfaces, 2017. **51**: p. 104-117.
11. Fenton, N. and J. Bieman, *Software metrics: a rigorous and practical approach*. 2014: CRC Press.
12. Lee, M.-C. and T. Chang, *Software measurement and software metrics in software quality.* International Journal of Software Engineering & Its Applications, 2013. **7**(4).
13. Padmini, K.J., H.D. Bandara, and I. Perera. *Use of software metrics in agile software development process*. in *Moratuwa Engineering Research Conference (MERCon), 2015*. 2015. IEEE.
14. Kan, S.H., *Metrics and models in software quality engineering*. 2002: Addison-Wesley Longman Publishing Co., Inc.
15. George, J.P., S. Abhilash, and K. Raja, *Transform domain fingerprint identification based on DTCWT.* Editorial Preface, 2012. **3**(1).
16. Basu, A., *Software Quality Assurance, Testing And Metrics*. 2015: PHI Learning Pvt. Ltd.
17. Nkiwane, N.H., W.G. Meyer, and H. Steyn, *The use of Earned Value Management for initiating directive project control decisions: A case study.* South African Journal of Industrial Engineering, 2016. **27**(1): p. 192-203.
18. Henard, C., et al. *Combining multi-objective search and constraint solving for configuring large software product lines*. in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. 2015. IEEE.
19. Banciu, D. and A. Balog, *Calitatea sistemelor si serviciilor de e-learning.* Editura AGIR, Bucuresti, 2013.
20. Ivan, I., et al., *Assigning Weights for Quality Software Metrics Aggregation.* Procedia Computer Science, 2015. **55**: p. 586-592.
21. Stefanoiu, D., et al., *Optimization in Engineering Sciences: Approximate and Metaheuristic Methods*. 2014: John Wiley & Sons.
22. Xue, W., L. Zhang, and X. Mou. *Learning without human scores for blind image quality assessment*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013.
23. Borne, P., et al., *Optimization in Engineering Sciences: Exact Methods*. 2013: John Wiley & Sons.

24. Fonseca, V.S., M.P. Barcellos, and R. Falbo. *Integration of Software Measurement Supporting Tools: A Mapping Study*. in *Twenty-Seventh International Conference on Software Engineering and Knowledge Engineering*. 2015.
25. Fonseca, V.S., M.P. Barcellos, and R. de Almeida Falbo, *Tools Integration for Supporting Software Measurement: A Systematic Literature Review*. iSys-Revista Brasileira de Sistemas de Informação, 2016. **8**(4): p. 80-108.
26. Czarnacka-Chrobot, B. *The ISO/IEC Standards for the Software Processes and Products Measurement*. in *SoMeT*. 2009.
27. Barcellos, M.P., R. de Almeida, and A.R. Rocha. *Establishing a well-founded conceptualization about software measurement in high maturity levels*. in *Quality of Information and Communications Technology (QUATIC), 2010 Seventh International Conference on the*. 2010. IEEE.
28. Arvanitou, E.-M., et al., *Software metrics fluctuation: a property for assisting the metric selection process.* Information and Software Technology, 2016. **72**: p. 110-124.
29. Mordal, K., et al., *Software quality metrics aggregation in industry*. Journal of Software: Evolution and Process, 2013. **25**(10): p. 1117-1135.
30. Fenton, N.E. and M. Neil, *Software metrics: roadmap*, in *Proceedings of the Conference on The Future of Software Engineering*2000, ACM: Limerick, Ireland. p. 357-370.
31. Ahmad, S.F., M.R. Beg, and M. Haleem, *A Comparative Study of Software Quality Models.* International Journal of Science, Engineering and Technology Research, 2013. **2**(1): p. 172.
32. Sharma, M., et al., *A comparative study of static object oriented metrics.* International Journal of Advancements in Technology, 2012. **3**(1): p. 25-34.
33. Chidamber, S.R. and C.F. Kemerer, *A metrics suite for object oriented design.* IEEE Transactions on software engineering, 1994. **20**(6): p. 476-493.
34. Sharma, M. and G. Singh, *Analysis of Static and Dynamic Metrics for Productivity and Time Complexity.* International Journal of Computer Applications (0975–8887) Volume, 2011.
35. Bivde, V. and P. Sarasu, *Correlation Between Coupling Metrics Values and Number of Classes in Multimedia Java Projects: A Case Study*. International Journal of Interactive Multimedia and Artificial Inteligence, 2016. **4**(Special Issue on Artificial Intelligence Underpinning).
36. Bowen, T., G. Wigle, and J. Tsai, *Specification of Software Quality Attributes. Volume 2. Software Quality Specification Guidebook*, 1985, DTIC Document.
37. Hammer, T.F., et al., *Doing requirements right the first time.* CROSSTALK The Journal of Defense Software Engineering, 1998: p. 20-25.
38. Janakiram, D. and M. Rajasree, *ReQuEst: Requirements-driven quality estimator.* ACM SIGSOFT Software engineering notes, 2005. **30**(1): p. 4.
39. Loconsole, A. *Measuring the requirements management key process area*. in *Proceedings of ESCOM-European Software Control and Metrics Conference, London, UK*. 2001.
40. Paulk, M.C., et al., *Key practices of the Capability Maturity Model version 1.1*. 1993.
41. Aliverdi, R., L.M. Naeni, and A. Salehipour, *Monitoring project duration and cost in a construction project by applying statistical quality control charts.* International Journal of Project Management, 2013. **31**(3): p. 411-423.
42. Menezes Jr, J., C. Gusmão, and H. Moura. *Indicators and metrics for risk assessment in software projects: a mapping study*. in *Proc. 5th Experimental Software Engineering Latin American Workshop (ESELAW 2008)*. 2008.
43. Rios, E., *Analise de riscos em projetos de Teste de software.* Editora Altabooks, 2005.
44. Cafeo, B.B.P., et al. *Towards indicators of instabilities in software product lines: An empirical evaluation of metrics*. in *2013 4th International Workshop on Emerging Trends in Software Metrics (WETSoM)*. 2013.
45. Molina-Castillo, F.J., et al., *Product Quality as a Formative Index: Evaluating an Alternative Measurement Approach.* Journal of Product Innovation Management, 2013. **30**(2): p. 380-398.
46. Selnes, F., *An examination of the effect of product performance on brand reputation, satisfaction and loyalty.* Journal of Product & Brand Management, 2013.
47. Yamada, S., *Software reliability modeling: fundamentals and applications*. Vol. 5. 2014: Springer.
48. Pham, H., *Quality and Reliability Management and Its Applications*. 2016: Springer.
49. Gillespie, A.M. *Reliability & maintainability applications in logistics & supply chain*. in *2015 Annual Reliability and Maintainability Symposium (RAMS)*. 2015. IEEE.

50. Hasim, N., N.A. Haris, and A.A. Rahman. *Defect Density: A Review of the Calculation Based on System Size*. in *Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication*. 2016. ACM.
51. Bansal, V. and P. Ahlawat, *A New Quality Model to Determine Quality of Product Built Using Object Oriented Concept.* Imperial Journal of Interdisciplinary Research, 2016. **2**(7).
52. Zhong, J.-H., et al., *Quantitative correlation between defect density and heterogeneous electron transfer rate of single layer graphene.* Journal of the American Chemical Society, 2014. **136**(47): p. 16609-16617.
53. Pfitscher, R.J., M.A. Pillon, and R.R. Obelheiro, *Customer-oriented diagnosis of memory provisioning for IAAS clouds.* ACM SIGOPS Operating Systems Review, 2014. **48**(1): p. 2-10.
54. Verhoef, P.C. and K.N. Lemon, *4. Advances in customer value management.* Handbook on Research in Relationship Marketing, 2015: p. 75.
55. Oliver, R.L., *Satisfaction: A behavioral perspective on the consumer*. 2014: Routledge.
56. Madeyski, L. and M. Jureczko, *Which process metrics can significantly improve defect prediction models? An empirical study.* Software Quality Journal, 2015. **23**(3): p. 393-422.
57. Rahman, F. and P. Devanbu. *How, and why, process metrics are better*. in *Proceedings of the 2013 International Conference on Software Engineering*. 2013. IEEE Press.
58. Abreu, F.B. and R. Carapuça. *Object-oriented software engineering: Measuring and controlling the development process*. in *Proceedings of the 4th international conference on software quality*. 1994.
59. Kanmani, S., et al., *Object-oriented software fault prediction using neural networks.* Information and software technology, 2007. **49**(5): p. 483-492.
60. Elish, K.O. and M.O. Elish, *Predicting defect-prone software modules using support vector machines.* Journal of Systems and Software, 2008. **81**(5): p. 649-660.
61. Gondra, I., *Applying machine learning to software fault-proneness prediction.* Journal of Systems and Software, 2008. **81**(2): p. 186-195.
62. Khoshgoftaar, T.M. and N. Seliya, *Comparative assessment of software quality classification techniques: An empirical case study.* Empirical Software Engineering, 2004. **9**(3): p. 229-257.
63. Gyimothy, T., R. Ferenc, and I. Siket, *Empirical validation of object-oriented metrics on open source software for fault prediction.* IEEE Transactions on Software engineering, 2005. **31**(10): p. 897-910.
64. Subramanyam, R. and M.S. Krishnan, *Empirical analysis of ck metrics for object-oriented design complexity: Implications for software defects.* IEEE Transactions on software engineering, 2003. **29**(4): p. 297-310.
65. Briand, L.C., et al., *Exploring the relationships between design measures and software quality in object-oriented systems.* Journal of systems and software, 2000. **51**(3): p. 245-273.
66. Graves, T.L., et al., *Predicting fault incidence using software change history.* IEEE Transactions on software engineering, 2000. **26**(7): p. 653-661.
67. Radjenović, D., et al., *Software fault prediction metrics: A systematic literature review.* Information and Software Technology, 2013. **55**(8): p. 1397-1418.
68. Gordieiev, O., V.S. Kharchenko, and M. Fusani. *Evolution of Software Quality Models: Green and Reliability Issues*. in *ICTERI*. 2015.
69. Eeles, P., *Capturing architectural requirements.* IBM Rational developer works, 2005.
70. Behkamal, B., M. Kahani, and M.K. Akbari, *Customizing ISO 9126 quality model for evaluation of B2B applications.* Information and software technology, 2009. **51**(3): p. 599-609.
71. Abran, A., et al., *Usability meanings and interpretations in ISO standards.* Software Quality Journal, 2003. **11**(4): p. 325-338.
72. Liu, M., et al. *Research on appraisal target system of software product quality metrics and evaluation*. in *2014 10th International Conference on Reliability, Maintainability and Safety (ICRMS)*. 2014.
73. Singh, B. and S.P. Kannojia. *A review on software quality models*. in *Communication Systems and Network Technologies (CSNT), 2013 International Conference on*. 2013. IEEE.
74. Samadhiya, D., W. Su-Hua, and C. Dengjie. *Quality models: Role and value in software engineering*. in *2010 2nd International Conference on Software Technology and Engineering*. 2010.
75. Ferenc, R., P. Hegedűs, and T. Gyimóthy, *Software Product Quality Models*, in *Evolving Software Systems*, T. Mens, A. Serebrenik, and A. Cleve, Editors. 2014, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 65-100.
76. Al-Qutaish, R.E., *Quality models in software engineering literature: an analytical and comparative study.* Journal of American Science, 2010. **6**(3): p. 166-175.

77. Rawashdeh, A. and B. Matalkah, *A new software quality model for evaluating COTS components.* Journal of Computer Science, 2006. **2**(4): p. 373-381.
78. Galin, D., *Software quality assurance: from theory to implementation*. 2004: Pearson Education India.
79. Ortega, M., M. Pérez, and T. Rojas, *Construction of a systemic quality model for evaluating a software product.* Software Quality Journal, 2003. **11**(3): p. 219-242.
80. Ronchieri, E., M.G. Pia, and F. Giacomini, *Software Quality Metrics for Geant4: An Initial Assessment.* the Proceedings of ANS RPSD, 2014: p. 14-18.

AUTHORS PROFILE

Anam Zai, is an aspiring software quality researcher, attained her BSCS from FUUAST Islamabad, Pakistan. Currently she is perusing MS degree in Software Engineering from International Islamic University Islamabad, Pakistan. Her research interests include (but not limited to) software quality assurance, software design & architecture, human computer interaction, and software requirements engineering.

Rawish Butt, is an innovative designer received her BSCS from FUUAST Islamabad, Pakistan. She is now a student of MS Software Engineering at International Islamic University Islamabad, Pakistan. Her research interests include mobile application design, software testing in the context of small mobile applications, model based software testing, and software quality assurance.

Shiza Nawaz, is a young researcher obtained her bachelor degree from UAJK, Pakistan. She is currently perusing MS degree in Software Engineering from International Islamic University Islamabad, Pakistan. She has a deep interest in research related to global software development, software quality engineering, wearable devices and agile methodology