Journal of Theoretical and Applied Information Technology
15th February 2019. Vol.97. No 3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SOFTWARE QUALITY MEASUREMENT IN SOFTWARE ENGINEERING PROJECT: A SYSTEMATIC LITERATURE REVIEW

[1]A.J. SUALI, *[2]S.S.M.FAUZI, [3]M.H.N.M. NASIR, [4]W.A.W.M. SOBRI, [5]I.K. RAHARJANA

[1, 2, 4]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA,

40450, Shah Alam, Selangor, Malaysia

[3]Faculty of Computer Science & Information Technology, University of Malaya,

50603 Kuala Lumpur, Malaysia

[5]Faculty of Science & Technology, Universitas Airlangga,

Surabaya, Indonesia

E-mail: *[2]shukorsanim@perlis.uitm.edu.my,

ABSTRACT

The quality of software satisfies when the requirements are fulfilled. The quality of software need to be measured to indicate the degree of satisfaction of the software to the users. There are several kinds of literature on software quality metrics have been published. However, very little research has been conducted to synthesise the measurement of software quality. The purpose of this paper is to synthesize the measurement of software quality in software engineering projects published in the literatures. Systematic Literature Review (SLR) was used to conduct the study. Systematic review in distinct stages was used such as the development of review protocol, search and keywords criteria, screening, development of inclusion and exclusion criteria, search for relevant studies, data extraction and synthesis. This study could give significant figures to measure software quality using different types of measurement such line of code (LOC), quality attribute, design pattern, the number of failure, fault, software trustworthiness, functional size, defect, and criteria software quality.

Keywords: *Software Quality, Systematic Literature Review, Measurement Type, Software Engineering, Software Quality Metrics*

1. INTRODUCTION

One of the main purpose of a software engineering project is to deliver high quality software [1]. The success of the software depends on whether it is delivered on time and within budget as well as maintaining a high quality. Focusing on improving software quality is important to the software developer who involve in the software development. Besides that, customer would concern on the quality software to satisfy their demand. An Institute of Electrical and Electronics Engineers (IEEE) standard is a standard used by software engineering committee to ensure the quality software could be fulfilled. The IEEE standard defines software as a set of data that instruct the computer to operate and avail the function to operate properly [2]. Meanwhile, quality is defined as the capability to accommodate the satisfaction of project needs or requirements.

As software become complex, more functions, flows and components are introduced to the software. For example, banking system has few functions at the early establishment such withdraw and transfer money, but the demand from customer to add new features makes the system become complex. Because of that, the software development processes becomes critical. This complexity of software development requires to be understood, studied and improved the quality product by measuring the quality to ensure the product achieved the expectations [3]. Measuring software quality is a tough task in terms of ensuring the software meets customers specification and needs

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

as every customer has their own definition of quality for the software [4].

The stakeholder from different division need to discuss together to reach the agreement which quality attributed they should achieved based on referring the software quality model [5]. Software quality model can be the guidelines for the practitioner to takes an action on improving the performance by measuring the quality software. There are several software quality model used in the software engineering such McCall model, Boehm model, Dromey model, FURPS model and ISO 9126 model [6] [7] [8] [9]. Each of the models carries different attributes that reflect to the quality product. Measuring the quality product becomes an essential element for the sake of success deliverable the software to the customer.

Measures the quality product gives beneficial to the organizational in term of provide information to support quantitative managerial decision-making during the systems development [10]. Prior literature suggest that measuring the quality of product such code review does deliver fewer post-release defects because of the frequent inspection after taken into consideration by the practitioner in every change [5]. The purpose of measuring the quality of software from the early development is to prevent the software fall into high risk and helps to find a solution to solve the problem. For instance, poor code review by the developers to inspect the defect in the codes has a negative impact to the software quality [11].

Other than that, prior study suggests poor designs and implementation methods lead into maintainability leak which reduces the quality of the software [12] [13]. Failure in measuring the quality product not only leaves bugs and errors but might cause over budgets, mission failure, injury and even lost a human life [14]. That is why the developers should prioritize to measure the quality product without making any mistakes. Other benefits measuring the quality software is to reach the predictable of performance and quality capability for ensuring the requirement achieved by the developers [15].

The different group of users may have different perspective of the quality of software [16]. This suggest, there are several ways to measure software quality [17] [18]. Unfortunately, very few studies have been conducted to synthesise the measurement used in software engineering projects. This paper aims to synthesise the measurement of software quality from previous studies. This study give an overall overview of the measurement types for software quality to the practitioner.

This paper is structured as follows, the next section discusses the methods used to conduct the study. Following that, results of the finding are discussed and software quality measurement is described. Finally, the conclusion summarises the whole paper.

2. METHODOLOGY

Systematic Literature Review (SLR) was employed to serve as a guideline in conducting the study. SLR supervises gathering of all papers from previous studies that are related to the topic areas and prevents repetition in collecting data [19]. SLR involves several phases which starts with the research question, search and keywords criteria, screening that is divided into inclusion and exclusion and finally the results and discussion. Figure 2 illustrates the process flow of the SLR.

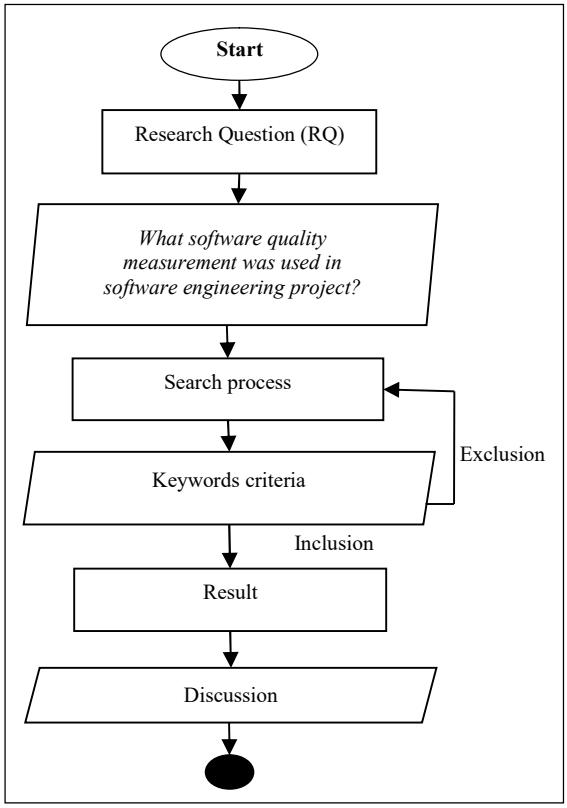

Figure 1: Flows of Systematic Literature Review

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2.1 Research Question

The following research question was used to guide the study:
RQ1: What software quality measurement was used in software engineering project?

2.2 Search and Keywords Criteria

The literature search used the keywords identified in Table 1. The keywords software quality measurement and software engineering project were combined using "OR" and "AND" Boolean operator.
The databases used in this study are stated as below:
- ACM Digital Library (http://dl.acm.org/)
- SciencemDirectnFreedomnCollection (http://www.sciencedirect.com/)
- IEEE Xplore (http://ieeexplore.ieee.org)
- Springer Online Journal Collection (http://link.springer.com/)
- Google Scholar (https://scholar.google.com/)

Table 1: Keywords Used In This Study

Category	Keywords
Software Quality Measurement	Software Quality Measurement/s
	Software Quality Metric/s
	Software Project Quality Measurement/s
	Software Project Quality Metric/s
	Project Quality Measurement/s
	Project Quality Metric/s
Software Engineering Project	Software Engineering
	Software Engineering Project/s
	Software Project/s
	Software Development
	Open Source Project
	Open Source Development

This study went back 30 years in time to distinguish how measurements of software quality were done throughout those years. The reason for this was to monitor the changes in measuring software quality throughout those years.

2.3 Screening Paper

Next, screening was done on the collected papers using inclusion and exclusion criteria to answer the research question. The following criteria of inclusion (I) and exclusion (E) were applied:
I1.Papers published directly mentioning the software quality measurement in the software project engineering or open source project.
I2.Papers which discuss software quality measurement.
E1.Posters, abstracts, article summaries and slide presentations.

Figure 1 shows the screening phases. During the first stage, the paper was screened from the title in which 121 papers were collected. After reading the abstract, only 76 papers were left out of 121 papers. After the introduction and content screening, 66 papers were left. Finally, the screening phase proceeds to reading the whole content of the paper and only 38 papers discussed on the software quality measurement in software engineering project.

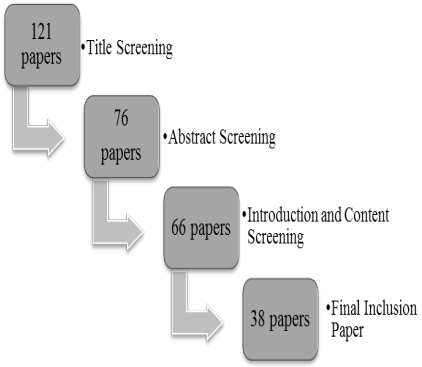

Figure 2: Screening Paper Phase

3. RESULT AND DISCUSSION

3.1 Overview of Studies

A total of 38 papers related to software quality measurement were selected for the study. Full list of paper can be seen in Appendix A. Table 2 illustrates the frequency of publication from 1984-2015 that is related to software quality measurement in software engineering project. Year 2014 displays the highest frequency of software quality measurement publication. This indicates the importance of software quality research in the software engineering field.

Table 2: Journal Tendency (By Year)

Years	Percentage (%)	Frequency
1984	3	1
1985	0	0
1986	0	0
1987	0	0
1988	0	0
1989	0	0
1990	3	1
1991	0	0
1992	0	0

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

1993	0	0
1994	5	2
1995	0	0
1996	0	0
1997	0	0
1998	3	1
1999	3	1
2000	0	0
2001	5	2
2002	0	0
2003	3	1
2004	8	3
2005	5	2
2006	3	1
2007	0	0
2008	5	2
2009	8	3
2010	10	4
2011	5	2
2012	5	2
2013	8	3
2014	13	5
2015	5	2
TOTAL	100	38

3.2 Study Classification

Software quality measurement was classified based on types of measurement. Table III illustrates several measurement types of software quality which include line of code (LOC), fault proneness, number of bug reports, source code, design pattern, criteria, fault data, and others. LOC showcases the highest percentage at 23 percent followed by defect and other measurement types.

Table 3: Types of Software Quality Measurement

Measurement Types	Percentage (%)	Frequency	Cite
LOC	32	12	[20] [17] [21] [22] [23] [24] [25] [26] [27] [28] [29] [30]
Quality Attribute	5	2	[31] [32]
Software Trustworthiness	5	2	[33] [18]
Design pattern	3	1	[34]
Number of failure	7	3	[35] [36] [37]
Fault	13	5	[38] [39] [40] [41] [42]
Functional size	3	1	[43]
Defect	16	6	[44] [45] [46] [47] [48] [49]
Criteria Software Quality	16	6	[50] [51] [52] [53] [54] [55]
Total	100	38	38

3.3 What Software Quality Measurement was used in Software Engineering Project (RQ1)?

The findings show that there were several kinds of measurement types discovered while gathering literature from the past 30 years. It includes Line of Code (LOC), quality attribute, design pattern, number of failure, fault, software trustworthiness, functional size, defect and criteria software quality.

3.3.1 Line of code (LOC)

In 1960, the first method to measure software quality was introduced and known as Line of Code (LOC) [27]. In software engineering projects, LOC can be calculated using Physical Lines, Logical Lines, Blank Lines, Total Lines of Code, Executable Physical, Executable Logical, Comment, Words in Comment, Header Comment, and Header Words [28].

Another technique to measure software quality using LOC is by reviewing [17] [22]. Reviewing focuses on analysing the code through documents, test plans and delivering prototypes to test the system function because failure could happen during development progress [22]. Computing LOC manually is a very difficult task and is time-consuming. Many tools have been developed to calculate LOC such as (SWMetrics) tool that uses Microsoft Visual Studio-C# to compute a metrics of LOC and Complexity based on the Cyclomatic quality measurement for many format languages using a source of code. This is determined by counting the number of basic paths through a function and calculated using the equation provided and control flow graph created from the equation [17]. UML based software is another tool used which is based on auto collected component metrics and predefined rules that will collect metrics automatically from generated software components that is used in the quality method [25].

3.3.2 Quality attribute

Quality attribute is divided into internal quality attribute and external quality attribute. Internal quality attribute uses quality criteria such as consistency, accuracy and testability as a result of quality factor decomposition [32]. Measuring internal quality involves factor, criteria and software metric that will form associations using

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

binary Boolean to improve quality by seeking evidence of the correlation between factor and criterion.

The relationship is the higher the correlation, the better the quality [32]. While external quality attribute can be measured consistently using scale measurement, the preference relation (such as that observed for a subjective ordinal scale) to validate expectation system for the attribute is relied on experts indicating that value provided by the assumption system informally agree with the experts intuition about the attribute [31].

3.3.3 Design pattern

Design pattern is another measurement of software quality. It has a remarkable impact on software development, maintenance, and reusability [34]. System design patterns can be framed as flows in painting and painters as a developer. Every part of the body in painting can be pointed as classes, interface and methods [34].

One method used to measure design pattern is Design Enhanced Quality Evaluation (DEQUALITE) method. DEQUALITE is a model to measure the quality of object-oriented systems that focus on internal attributes and design. The design involves five steps, starting with choosing appropriate characteristics, identifying and listing the most significant, tangible, internal attributes of systems implementing design patterns, assessing system attributes, evaluating the impact of design patterns on quality and finally carrying out a validation and a refinement of the resulting quality model [34].

3.3.4 Number of failure

Measuring number of failures is another way to measure software quality. Failure can be reflected in the system itself where the system fails to operate based on requirements required and delivers wrong results compared to expected results [37]. STREW-J metric suite is a method to detect internal failure at an early stage of development to improve the quality [35]. STREW-J metric suite uses test suite to measure failure, but if the test suite is not applicable, the tester would use historical data from a comparable project [35].

Test suite is a bunch of test cases that is compressed into one test suite. A test suite that contains many test cases needs to be run into selected programs such as SoapUI to assess whether the actual result matches the expected output and test suite might be a fundamental part of the software design [36]. Test suite can predict the number of failures once results are obtained.

3.3.5 Fault

Fault is a software defect that causes failure, and it is another way to measure software quality by counting the numbers of defects during software project development [38]. There are a few techniques that can be used to measure fault. One of the approaches is to use prediction to predict the occurrence of faults by applying a technique known as semi-supervised clustering approach and EM-based approach [41] [42] [39].

3.3.6 Software trustworthiness

Software trustworthiness evaluation goal is to help maximize the area of quality. The definition of software trustworthiness is the degree of confidence in a set of requirements that include functional and non-functional requirements [33]. Software trustworthiness alerts on the security of software products and demands for action and state to be under control during all stages of development [18].

Since software trustworthiness covers all stages of life-cycle, the measurement will start from requirement procedure trustworthiness through satisfaction, measurement designing procedure trustworthiness through internality degree metric of functions, measurement coding procedure trustworthiness through validity measurement of codes, measurement testing procedure through error risk measurement and every stage will be measured to identify the degree of satisfaction [18].

3.3.7 Functional Size

Quality software can also be measured using Functional Size Measurement (FSM) which is quite important in the current software development practice. This is because most effort estimation models rely on evaluation of application sizes to be developed. Function points, meant to consider the customer's point of view in a technology independent manner, are a measure of functional requirements that take into account elements that can be identified by the user [43].

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

FPs is based on five Basic Functional Components (BFCs): External Input , External Output, External Inquiry , Internal Logical File and External Interface File.

3.3.8 Defect

Wrong results are obtained when incorrect action, operation and data happen. Bugs usually found in codes can be defined as a defect [37]. Finding and fixing the defect during the early stage of development will save a lot of budget and development time. Analyzing the defect early on can improve software quality but increase cost, which means quality assessments need to be done often and hence, are elaborate procedures [45].

The most cost-effective early defect detection technique as suggest in prior literature is "Fagan Inspection" [44]. Fagan inspection chooses the number of defects in an element as a response variable, and (depending on the hypothesis) measurements of such factors like complexity of the element, the number of contributors, the development organization's social and geographic structure, as predictor variables. Defect data is mined and used with machine learning models to automatically predict likely future loci for defects. The data frequently suffers from class imbalance.

3.3.9 Criteria of software quality

In a development project, certain criteria such as reliability, maintainability, usability, functionality and efficiency are the criteria to be fulfilled during this project [55]. Every measurement needs certain preconditions to measure software quality. In an Object-oriented (OO) program, prediction of quality is usually based on values of the criteria. Positive values of the quality criteria will ensure an OO program of higher quality [50]. Indirect measurement of an attribute involves the measurement of one or more attributes.

Unpredictable behaviour with adverse effects might happen when there is faulty implementation of the context-aware features. Context diversity is to capture the extent of context changes in serial inputs and to propose three strategies to study how context diversity may improve the effectiveness of data-flow testing criteria. Software measurement is defined as a system which includes all aspects of software measurement, evaluation, estimation and exploration [51].

If criteria were the cause of reported failure, continuing to use the same criteria will simply repeat failures of the past. The process criteria measurement for project management is measuring efficiency [52]. Criteria are defined as measuring and interpreting conformance with quality requirements during inspection and testing [53]. The overall quality grade depends on the knowledge-based importance of characteristics [54].

Table 4: Metrics for Object Oriented Systems

Source	Metric	OO Construct
Traditional	Cyclomatic complexity (CC)	Method
	Lines of Code(LOC)	Method
	Comment Percentage (CP)	Method
CK Object Oriented Metric	Weighted Methods per Class (WMC)	Class/Method
	Response for a Class (RFC)	Class/Method
	Lack of Cohesion of Methods (LCOM)	Class/Method
	Coupling Between Objects (CBO)	Coupling
	Depth of Inheritance Tree (DIT)	Inheritance
	Number of Children (NOC)	Inheritance

Table 4 shows the metric for an object-oriented system that is used for measuring quality. Every characteristic of software quality is measured to ensure the quality meets the standard.

In the first stage of measuring software quality, components of software measurement will be considered at different levels of each component in order to classify different levels of software measurement. It is important to identify the measurement obtained at the early phase of the life circle. Figure 2 shows the evaluation measurement that happens in all stages of development starting with early measurement in the analysing phase which is the documentation observation until the testing and operation phase, which involves codes to be executed [53].

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

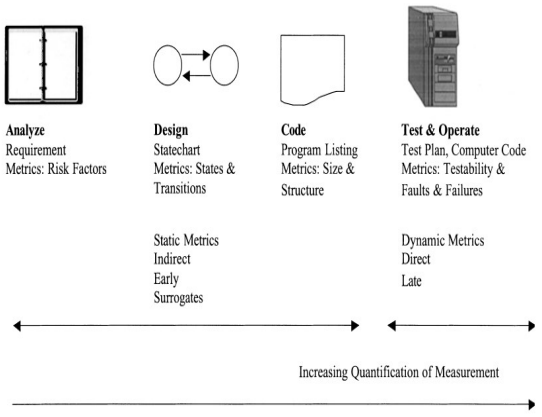

Figure 3: Life cycle measurement attributes [53]

The fundamental process in software quality measurement is to consider the important criteria of products before evaluating the quality of products. In order to measure software quality, several general criteria must be listed so that measurement can be carried out. To identify the general criteria, a survey was conducted among customers and end users using scale to identify the quality which demands to be measured [54].

4. CONCLUSION

This paper applies Systematic Literature Review (SLR) to synthesise the measurement of software quality that is used in software engineering projects in previous literature. Nine software quality measurements in software engineering projects were identified which includes line of code (LOC), quality attribute, design pattern, number of failure, fault, software trustworthiness, functional size, defect, and criteria software quality. One of the measures identified will be used as a measure of software quality in Socio-Technical Congruence (STC) study.

4. ACKNOWLEDGEMENTS

We are grateful to the Ministry of Higher Education for supporting this research, through the Research Acculturation Grant Scheme (RAGS) grant.

REFERENCES:

[1] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, A. Folleco, "An empirical study of the classification performance of learners on imbalanced and noisy software quality data", *Information Sciences*, Vol. 259, 2014, pp. 571–595

[2] Radatz, Jane, A. Geraci, F. Katki, "IEEE Standard Glossary of Software Engineering Terminology", *IEEE Std 610121990.121990*, vol. 12, 1990, pp. 3.

[3] R. Pressman, "Software Process Impediment," *IEEE software*, Vol. 13, No. 5, 1996, pp. 16–17.

[4] M. Khaliq, R. A. Khan, M. H. Khan, "Software quality measurement : A Revisit, Oriental", *Journal of Computer Science & Technology*, Vol. 3, No. 1, 2010, pp. 5–11.

[5] S. McIntosh, Y. Kamei, B. Adams, A. E. Hassan, "The impact of code review coverage and code review participation on software quality: a case study of the qt, VTK, and ITK projects," *Proceedings of the 11th Working Conference on Mining Software Repositories*, 2014, pp. 192–201.

[6] T. Davuluru, J. Medida, V. S. K. Reddy, "A study of software quality models," *Advances in Engineering and Technology Research (ICAETR), 2014 International Conference on*, 2014, pp. 1-8.

[7] M. Ortega, M. Pérez, T. Rojas, "Construction of a systemic quality model for evaluating a software product," *Software Quality Journal*, Vol. 11, No. 3, 2003, pp. 219–242.

[8] A. B. Al-badareen, M. H. Selamat, M. A. Jabar, "Software quality models: A comparative study," *International Conference on Software Engineering and Computer Systems*, 2011, pp.46-55.

[9] J. P. Miguel, D. Mauricio, G. Rodríguez, "A Review of Software Quality Models for the Evaluation of Software Products," *arXiv preprint arXiv:1412.2977*, 2014, pp. 31–53.

[10] L. G. Wallace, S. D. Sheetz, "The adoption of software measures: A technology acceptance model (TAM) perspective," *Information & Management*, Vol. 51, No. 2, 2014, pp. 249–259.

[11] S. McIntosh, Y. Kamei, B. Adams, A. E. Hassan, "An empirical study of the impact of modern code review practices on software quality," *Empirical Software Engineering*, Vol. 21, No. 5, 2016, pp. 2146–2189.

[12] M. Tufano, "When and Why Your Code Starts to Smell Bad.," *Proceedings of the 37th International Conference on Software Engineering-Volume 1*, 2015, pp. 403–414.

[13] R. Marinescu, "Measurement and quality in object-oriented design," in *Software Maintenance, 2005. ICSM'05. Proceedings of*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

the 21st IEEE International Conference on, 2005, pp. 701–704.

[14] D. Jamwal, "Analysis of Software Quality Models for Organizations," *International Journal of Latest Trends in Computing*, Vol. 1, No. 2, 2010, pp. 19–23.

[15] M. Unterkalmsteiner, T. Gorschek, A. K. M. M. Islam, C. K. Cheng, R. B. Permadi, R. Feldt, "Evaluation and measurement of software process improvement-A systematic literature review," *IEEE Transactions on Software Engineering*, Vol. 38, No. 2, 2012, pp. 398–424.

[16] D. Chappell, "The Three Aspects of Software Quality: functional, structural, and process", *in Retrieved from*, 2013.

[17] S. Zakariya, M. Belal, "Software Quality Management Measured Based Code Assessments", *International Journal of Computer Science Trends and Technology (IJCST)*, Vol. 3, No. 4, 2015, pp. 263–268.

[18] B. Yu, Q. Wang, Y. Yang, "The Trustworthiness Metric Model of Software Process Quality Based-on Life Circle", *in Management and Service Science, 2009. MASS'09. International Conference on*, 2009, pp. 1–5.

[19] Kitchenham, B., S. Charters, "issue: EBSE 2007-001.", *ISBN: 1595933751*, 2007.

[20] L. Crispin, "Driving software quality: How test-driven development impacts software quality", *IEEE Software*, Vol. 23, No. 6, 2006, pp. 70–71.

[21] P. Thongtanunam, R.G. Kula, A. E. C. Cruz, N. Yoshida, H. Iida, "Improving code review effectiveness through reviewer recommendations", *in Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering - CHASE 2014*, 2014, pp. 119–122.

[22] G.M. Weinberg, D.P. Freedman, "Reviews, Walkthroughs, and Inspections", *IEEE Transactions on Software Engineering*, Vol. 1, 1984, pp. 68–72.

[23] M. Huo, J. Verner, L. Zhu, M.A. Babar, "Software quality and agile methods, *in Computer Software and Applications Conference, 2004. COMPSAC 2004". Proceedings of the 28th Annual International*, 2004, pp. 520–525.

[24] S. Mcintosh, Y. Kamei, B. Adams, A.E. Hassan, "The Impact of Code Review Coverage and Code Review Participation on Software Quality Categories and Subject Descriptors", *in Proceedings of the 11th Working Conference on Mining Software Repositorie (MSR 2014)*, 2014, pp. 192–201.

[25] B. Deniz, "Software Component Score: Measuring Software Component Quality Using Static Code Analysis", *Computational Science and Its Applications -ICCSA*, Vol. 5, 2015, pp. 63–72.

[26] K. Hotta, Y. Sano, Y. Higo, S. Kusumoto, "Is Duplicate Code More Frequently Modified Than Non-duplicate Code in Software Evolution?: An Empirical Study on Open Source Software", *in Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE)*, 2010, pp. 73–82.

[27] C. Jones, "A Short History of the Lines of Code (Loc) Metric", *Capers Jones & Associates LLC, Narragansett,* 2008, pp. 1-12.

[28] K. Bhatt, V. Tarey, and P. Patel, "Analysis Of Source Lines Of Code (SLOC) Metric", *International Journal of Emerging Technology and Advanced Engineering*, Vol. 2, No. 5, 2012, pp. 150–154.

[29] M. Beller, A. Bacchelli, A. Zaidman, and E. Juergens, "Modern code reviews in open-source projects: which problems do they fix?", *in Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, 2014, pp. 202–211.

[30] A. Cockburn, L. Williams, "The costs and benefits of pair programming, Extreme programming examined", *Inc., Boston, MA*, 2001, pp. 223–243.

[31] J. Moses, M. Farrow, "Tests for consistent measurement of external subjective software quality attributes, Empirical Software Engineering", *Empirical Software Engineering*, Vol. 13, No. 3, 2008, pp. 261–287.

[32] W. Pedryez, J.F. Peters, S. Ramanna, "Software Quality Measurement: Concepts and Fuzzy Neural Relational Model", *in Fuzzy Systems Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, 1998, pp. 1026–1031.

[33] E. Amoroso, C. Taylor, J. Watson, J. Weiss, "A Process-Oriented Methodology for Assessing and Improving Software Trustworthiness", *in Proceedings of the 2nd ACM Conference on Computer and communications security*, 1994, pp. 39–50.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[34] F. Khomh, Y.G. Guéhéneuc, "DEQUALITE : Building Design-based Software Quality Models", *in Proceedings of the 15th Conference on Pattern Languages of Programs*, 1994, p. 2.

[35] N. Nagappan, L. Williams, M. Vouk, J. Osborne, "Early estimation of software quality using in-process testing metrics", *in International Conference on Software Engineering: Software Quality conference proceedings*, 2005, pp. 1-7.

[36] E. Volokh, "Test Suites : A Tool for Improving Student Articles", *Journal Legal Education*, Vol. 52, No. 3, 2003, pp. 440-445.

[37] E. E. Ogheneovo, "Software Dysfunction : Why Do Software Fail?", *Journal of Computer and Communications*, Vol. 2, No. 6, 2014, pp. 25–35.

[38] N. Seliya, T. M. Khoshgoftaar, S. Zhong, "Analyzing software quality with limited fault-proneness defect data", *in Ninth IEEE International Symposium on High-Assurance Systems Engineering (HASE'05)*, 2005, pp. 89–98.

[39] R. Malhotra, A. Jain, "Fault Prediction Using Statistical and Machine Learning Methods for Improving Software Quality", *Journal of Information Processing Systems*, Vol. 8, No. 2, 2012, pp. 241–262.

[40] N. Seliya, Naeem, T.M. Khoshgoftaar, "Semi-Supervised Learning for Software Quality Estimation", *in Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on, 2004*, pp. 183–190.

[41] S. Zhong, T. M. Khoshgoftaar, N. Seliya, "Unsupervised Learning for Expert-Based Software Quality Estimation", *in Proceedings of the Eighth IEEE International Symposium on High Assurance Systems Engineering (HASE'04)*, 2004, pp. 149–155.

[42] C. Catal, "Software fault prediction: A literature review and current trends, Expert systems with applications", *Expert systems with applications*, Vol. 38, No. 4, 2011, pp. 4626–4636.

[43] L. Lavazza, G. Robiolo, "The Role of the Measure of Functional Complexity in Effort Estimation", *in Proceedings of the 6th International Conference on Predictive Models in Software Engineering*, 2010, pp. 1–10.

[44] M. Bush, "Improving Software Quality: The Use of Formal Inspections at the Jet Propulsion Laboratory", *in Software Engineering, 1990. Proceedings*, 12th International Conference on, 1990, pp. 196–199.

[45] S. Wagner, "The Use of Application Scanners in Software Product Quality Assessment", *in Proceedings of the 8th international workshop on Software quality*, 2011, pp. 42–49.

[46] J. Li, N. B. Moe, T. Dyba, "Transition from a Plan-Driven Process to Scrum – A Longitudinal Case Study on Software Quality", *in Proceedings of the 2010 ACM-IEEE international symposium on empirical software engineering and measurement*, 2010, pp. 13.

[47] D. Athanasiou, A. Nugroho, J. Visser, A. Zaidman, "Test Code Quality and Its Relation to Issue Handling Performance", *IEEE Transactions on Software Engineering*, Vol. 40, No. 11, 2014, pp. 1100–1125.

[48] M. Staron, W. Meding, M. Caiman, E. Ab, "Improving Completeness of Measurement Systems for Monitoring Software Development Workflows", *in Systems for Monitoring Software Development Workflows*, 2013, pp. 230–243.

[49] E. Kouroshfar, M. Mirakhorli, H. Bagheri, L. Xiao, S. Malek, Y. Cai, "A Study on the Role of Software Architecture in the Evolution and Quality of Software", *in Proceedings of the 12th Working Conference on Mining Software Repositories*, 2015, pp. 246–257.

[50] C. Suite, R. Kumar, "Indirect method to measure software quality using CK-OO suite, in Intelligent Systems and Signal Processing (ISSP)", *2013 International Conference*, 2013, pp. 47–51.

[51] R. Dumke, H. Yazbek, E. Asfoura, K. Georgieva, "A General Model for Measurement Improvement", *in Software Process and Product Measurement*, 2009, pp. 48–61.

[52] R. Atkinson, "Project management: cost time and quality two best guesses and a phenomenon, it's time to accept other success criteria", *International journal of project management*, Vol. 17, No. 6, 1999, pp. 337–342.

[53] N.F. Schneidewind, "Knowledge Requirements for Software Quality Measurement", *Empirical Software Engineering*, Vol. 6, No. 3, 2001, pp. 201–205.

[54] R. Hofman, "An Approach to Measuring Software Quality Perception", *in Innovations in Computing Sciences and Software Engineering*, 2010, pp. 307–312.

[55] A. Abran, R.E. Al-Qutaish, J.M. Desharnais, N. Habra, "ISO-Based Models To Measure Software Product Quality", *in Institute of Chartered Financial Analysts of India (ICFAI)-*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ICFAI Books, 2008, pp. 61–96.

[56] N. Nagappan, L. Williams, M. Vouk, J. Osborne, "Early estimation of software quality using in-process testing metrics," *ACM SIGSOFT Software Engineering Notes*, Vol. 30, No. 4, 2005, pp. 1-7.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

APPENDIX A: LIST OF PAPER

Citation	Author	Year	Title
[20]	L. Crispin	2006	Driving software quality: How test-driven development impacts software quality
[17]	S. Zakariya and M. Belal	2015	Software Quality Management Measured Based Code Assessments
[21]	P. Thongtanunam, R. G. Kula, A. E. C. Cruz, N. Yoshida, and H. Iida	2014	Improving code review effectiveness through reviewer recommendations
[22]	G. M. Weinberg and D. P. Freedman	1984	Reviews, Walkthroughs, and Inspections
[23]	M. Huo, J. Verner, L. Zhu, and M. a Babar	2004	Software quality and agile methods
[24]	S. Mcintosh, Y. Kamei, B. Adams, and A. E. Hassan	2014	The Impact of Code Review Coverage and Code Review Participation on Software Quality Categories and Subject Descriptors
[25]	D. Triantakonstantis and S. Barr	2009	Computational Science and Its Applications -ICCSA
[26]	K. Hotta, Y. Sano, Y. Higo, and S. Kusumoto	2010	Is Duplicate Code More Frequently Modified Than Non-duplicate Code in Software Evolution?: An Empirical Study on Open Source Software
[27]	C. Jones	2013	A Short History of the Lines of Code (Loc) Metric
[28]	K. Bhatt, V. Tarey, and P. Patel	2012	Analysis Of Source Lines Of Code (SLOC) Metric
[29]	M. Beller, A. Bacchelli, A. Zaidman, and E. Juergens	2014	Modern code reviews in open-source projects: which problems do they fix?
[30]	A. Cockburn and L. Williams	2001	The costs and benefits of pair programming
[31]	J. Moses and M. Farrow	2008	Tests for consistent measurement of external subjective software quality attributes
[32]	S. Ramanna	1998	Software Quality Measurement: Concepts and Fuzzy Neural Relational
[33]	E. Amoroso, C. Taylor, J. Watson, and J. Weiss	1994	A Process-Oriented Methodology for Assessing and Improving Software Trustworthiness 2 Definition of Software Trustworthiness Trust Principles
[18]	B. Yu, Q. Wang, and Y. Yang	2009	The Trustworthiness Metric Model of Software Process Quality Based-on Life Circle
[34]	F. Khomh	1994	DEQUALITE : Building Design-based Software Quality Models
[56]	N. Nagappan, L. Williams, M. Vouk, and J. Osborne	2005	Early estimation of software quality using in-process testing metrics
[36]	E. Volokh	2003	Test Suites : A Tool for Improving Student Articles
[37]	E. E. Ogheneovo	2014	Software Dysfunction : Why Do Software Fail?
[38]	N. Seliya, T. M. Khoshgoftaar, and S. Zhong	2005	Analyzing software quality with limited fault-proneness defect data
[39]	R. Malhotra and A. Jain	2012	Fault Prediction Using Statistical and Machine Learning Methods for Improving Software Quality
[40]	N. Seliya	2004	Semi-Supervised Learning for Software Quality Estimation
[41]	E. Ieee, I. Symposium, H. Assurance, and S. Engineering	2004	Unsupervised Learning for Expert-Based Software Quality Estimation
[42]	C. Catal	2011	Software fault prediction: A literature review and current trends
[43]	L. Lavazza and G. Robiolo	2010	The Role of the Measure of Functional Complexity in Effort Estimation
[44]	M. Bush	1990	Improving Software Quality: The Use of Formal Inspections at the Jet Propulsion Laboratory
[45]	S. Wagner	2011	The Use of Application Scanners in Software Product Quality Assessment
[46]	J. Li, N. B. Moe, and T. Dyba	2010	Transition from a Plan-Driven Process to Scrum - A Longitudinal Case Study on Software Quality
[47]	D. Athanasiou, A. Nugroho, J. Visser, and A. Zaidman	2014	Test Code Quality and Its Relation to Issue Handling Performance
[48]	M. Staron, W. Meding, M. Caiman, and E. Ab	2013	Improving Completeness of Measurement Systems for Monitoring Software Development Workflows
[49]	E. Kouroshfar, M. Mirakhorli,	2015	A Study on the Role of Software Architecture in the Evolution

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

	H. Bagheri, L. Xiao, S. Malek, and Y. Cai		and Quality of Software
[50]	C. Suite and R. Kumar	2013	Indirect Method to Measure Software Quality using
[51]	R. Dumke, H. Yazbek, E. Asfoura, and K. Georgieva	2009	A General Model for Measurement Improvement
[52]	R. Atkinson	1999	Project management: cost time and quality two best guesses and a phenomenon, it's time to accept other success criteria
[53]	N. F. Schneidewind	2001	Knowledge Requirements for Software Quality Measurement
[54]	R. Hofman	2010	An Approach to Measuring Software Quality Perception
[55]	A. Abran, R. E. Al-Qutaish, J.-M. Desharnais, and N. Habra	2008	Iso-Based Models To Measure Software Product Quality