I.J. Information Technology and Computer Science, **2022, 6, 13-22**
Published Online on December 8, 2022 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2022.06.02

Evaluation of Software Quality in Test-driven Development: A Perspective of Measurement and Metrics

Ikenna Caesar Nwandu*
Department of Software Engineering, Federal University of Technology, Owerri Nigeria
E-mail: ikenna.nwandu@futo.edu.ng
ORCHID iD: https://orcid.org/0000-0001-6834-1088
*Corresponding Author

Juliet N. Odii
Department of Computer Science, Federal University of Technology, Owerri Nigeria
E-mail: juliet.odii@futo.edu.ng

Euphemia C. Nwokorie
Department of Computer Science, Federal University of Technology, Owerri Nigeria
E-mail: euphemia.nwokorie@futo.edu.ng

Stanley A. Okolie
Department of Computer Science, Federal University of Technology, Owerri Nigeria
E-mail: sokolie@futo.edu.ng

Received: 11 November 2021; Revised: 22 June 2022; Accepted: 14 September 2022; Published: 08 December 2022

Abstract: A software product is expected to be subjected to critical evaluation on its quality attributes in order to ascertain that target quality requirements are met, and that those quality attributes responsible for revealing software quality are not omitted in the software development process. Software metrics are essential to accomplish the task. This paper has carried out an exploratory study of software measurement and software metrics in tandem. The study took into cognizance the interwoven nature of the duo in measuring and revealing software quality. The study formulated a model that expressed the mutual bonding that propels both measurement and metrics to describing software quality in numeric quantities of software attributes. The study identified six software attributes whose values are considered enough quantities to reveal the quality of a software product. The identification enabled the study to create a model equation aimed at giving a numeric value for the complete evaluation of a software system. The result of the implementation of the six software attributes into the model equation showed that two software products employed in the study are of high-quality, having quality values of 0.93 and 0.86 respectively. The attributes produced values that confirmed the maintainability (25 seconds & 20 seconds respectively) and reliability (0.78 & 0.80 respectively) of both software products among other differing features that characterize them.

Index Terms: Software Evaluation, Software Measurement, Software Metrics, Software Quality, Software Attributes.

1. Introduction

Software quality is a resultant outcome of the quantitative investigation of the extent at which certain quality attributes are domiciled within a software product. The quality of software products attracts much relevance in our present world. This is as a result of the unending pervasiveness of software products which is driven by the daily advancement in information technology whose importance is on the increase in the post-pandemic period. That is why software quality must be mapped and investigated to embrace correctness and precision. A good software development process accommodates this investigative process which is usually described as software evaluation. Software evaluation is a necessary activity that validates the efficacy of development at every stage of the software process. Software evaluation is therefore an unprecedented task that ensures maintenance of standardized methodologies in implementing

software requirements, specifications, design, project management, and quality assurance measures. The implementation of these software engineering activities confirms the assurance of the reliability of a software product. The strategy of operation through which a software product is developed is therefore, a factor to be considered. A four-step procedure was formulated by [1] with the intent of assisting in the measurement of software product quality. The sequence of activities that make up the procedure is as follows:

- State the scope of the quality
- State the limits and building blocks for the quality metrics
- State the modalities for measurement
- Perform measurement with respect to the quality metrics.

The method or technique used for evaluation notwithstanding, the process is expected to unleash the attributes that are bound to the overall software quality. This study made effort to positioning software evaluation towards making use of the most relevant attributes to express software quality. This is necessary owing to the cumbersomeness of involving all known attributes in the investigative process of software evaluation. A total of six software attributes (namely reliability, usability, efficiency, functionality, maintainability and portability) was objectively identified to reasonably serve the purpose of quality evaluation regardless of numerous others. Reliability of software is a check on the software correctness. It tries to give assurance that the software does not fail over a period of time within a specified working platform. Usability, on similar note, determines what, when, where, and how a software can perform effectively. This makes the attribute essential considering the foundational properties it considers. Efficiency of software is involved with the behaviour of the system over time in relation to resources. It is considered important because it varies actual results with some system estimated values in order to determine the system performance. Functionality is an indication of how accurate and adequate a software can perform. It therefore shows the level of system compliance to specifications. Maintainability of software is its ability to be easily modified. This creates room for improvement and robustness. Portability attribute ensures that there is effective execution of same software on diverse platforms. That is why it also forms a basic concern. This study further modelled the relationship between metrics and measurement to form an intersecting boarder which illustrates the quality of software.

The study is divided into seven sections. Section 1 is the background information which introduced the phenomena of metrics and measurement as regards software development, section 2 presented the relationship that binds metrics and measurement into an interwoven model, section 3 gave a couple of literatures indicating some existing studies that have relationships with software evaluation using metrics and measurements, section 4 presented a model equation formulated to implement the evaluation vis-à-vis the attributes employed in the study to produce the supposed software quality value, section 5 presented the study method which identified the selected software attributes and why they are considered key in evaluating software quality, section 6 presented the results gotten from the deployment of the various metrics that help to create values for the software attributes, and finally, section 7 presented the conclusions drawn from the findings of the study.

2. Metrics-measurement Bonding Relationship

Metrics and measurement are basic evaluation tools that are widely put into use by both industry actors and academic scholars in the course of determining and expressing software attributes in quantified values. Metrics are perceived to be directly related to the functions whereas measurement refers to the outcomes of the functions. In other words, metrics are the results of the computational formulas used in deriving outcomes that are seen as the measurements.

2.1. Metrics

Metrics are indicators that sights the internal aspect of software quality [2]. In software engineering, metrics are of three main categories - *product metrics*, *process metrics* and *project metrics*.

Product metrics are the dynamic tools that quantify the internal features of a software with the intent of expressing the level of the software's conformation to specifications [3]. They are however called predictor metrics since their operations gear towards estimating the functionality of the software product. This category of metrics is further grouped into dynamic and static metrics according to the properties of the software product in which they try to measure. Metrics that belong to the dynamic class are known to express the dynamic behavioural display of software execution. For instance, the time it takes a software to complete a computation, or the rate at which a software experiences an interrupt can be classified as dynamic metrics [4]. Such metrics usually give preference to the evaluation of software efficiency and reliability statuses while keeping other attributes to dormancy. On the other hand, static metrics considers to express the characteristic properties of software products. By so doing, they evaluate how simple, understandable, complex or maintainable the software can be. Examples of static metrics include lines of code (LOC), cyclomatic complexity, and fog-index.

Process metrics are tools for determining the rate of progress recorded in software development process. They further investigate the characteristics of deployed methodologies that would accomplish the implementation of the

desired properties into the software. Process metrics are responsible for the determination of software process, the techniques needed to execute the various activities, and the tools needed to implement each stage of the process. The operations of process metrics make them the determinants of the organisation's policy, such that the decision on whether or not a deployed software process is retained is highly dependent on the feedback gotten from process metrics. Process metrics therefore play the role of ascertaining the efficacy of a software process for a particular type of software to be developed. This means that an organization and her development team get the idea of the effectiveness of a particular process adopted for a project via process metrics [5].

Project metrics are important for evaluating the software process ability to manage the resources made available to it. Process metrics perform this evaluation to reveal how the process resources are utilized and the impact they make on the development process. They help to describe the characteristics of the project as well as the software attributes at every level of development and further assist in mapping out execution plan [6,7]. This class of metrics therefore guide the development team in rating their efforts. They also allow organisation's management to analyse the progress of the process, the cost and time expended and the estimated cost and time of the project. This means that project metrics provide a control on software development activities [8] which are necessary for the assurance of software that has good quality. Ref. [9] described it as management metrics, invariably refers to statistical software quality assurance data. Process metrics ensures that there is consistency in expected period of delivery, costs estimation, project schedule, man-hour and all deliverables of the project.

Table 1 gives a summary of the importance of metrics in the business of evaluating software products with respect to the various types and their relevant examples.

Table 1. Metrics and their Importance

Metric Category	Importance	Examples
Product Metrics ***(Dynamic, Static)***	(i) Evaluates software efficiency and reliability *(dynamic)* (ii) Expresses software characteristics *(static)*	*Dynamic*: (i) Computation completion time (ii) Interrupt occurrence rate *Static*: (i) Lines of code (LOC) (ii) Cyclomatic complexity (iii) Fog-index
Process Metrics	(i) Determines organization's policy to retain or discard software (ii) Ensures implementation of desired properties	(i) Efficacy analysis result (ii) Stage-wise feedback
Project Metrics	(i) Reveals utilization and impartation rate of resources (ii) Controls overall software development	(i) Delivery time (ii) Cost estimation (iii) Project schedule (iv) Man-hour

Table 2. Software attributes and standards of measurements

Attributes	Standard Measures
Reliability	Cyclomatic complexity Programs' lines of code Fault tolerance
Usability	Length of users' manual. Number of error messages. Capability to be understood, learned, and used (ISO/IEC 9126-1, 2000)
Efficiency	Device efficiency Accessibility Correctness
Functionality	Accuracy Adequacy Interoperability Compliance to specifications
Maintainability	Readability Reusability Extensibility Understandability
Portability	Installability Adaptability Compatibility

Source: [5,18]

2.2. Measurement

Measurement refers to ascribing values to real life software attributes to express them with some laid down standards [10]. Hence, it provides visibility of system functionality [2]. This explains why measurement cannot be relegated to the background within the test-driven development circle. This is because test-driven development lays emphasis on the tangible value of specific attributes which serves as evaluation information. Measurement, as a process, maintains a check on the resultant software product vis-à-vis its objectives and scope of functionality. Hence, it is a

subjective process which aims at some specific attributes of software. Measurement therefore ensures that the characteristics of those target attributes are properly harnessed. Measurement reveals the physical and abstract properties of a software product via the evaluation of its attributes. In software engineering, different software attributes are measured in standardized formats (see table 2). These standards provide means of exposing the differences between various forms of software systems. The exposure is a way of determining the characteristics inherent in a particular software product and how those characteristics influence the system performance.

2.3. Metric-measurement Bond

The standards in which software attributes are measured are called metrics. Hence, a metric can be further described as a function of measurement which ascribes value to a software attribute. This implies that software measurement is an outcome of putting software metrics into use. This explains why we say that software metrics are the formulas whereas software measurement are the values derived from the formulas. Therefore, the relationship between metrics and measurement obviously lies in the complimentary expression of the duo in quantifying the attributes that describe software quality. The relationship seems to be inseparable since the absence of one suggests the absence of all as depicted in fig. 1. The relationship makes an intersection that forms the quality of software while describing measurement as a subset of a broader function called metrics.

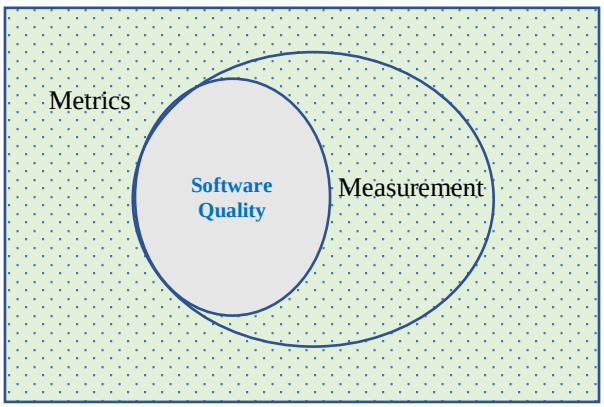

Fig.1. Bonding Relationship of Metrics and Measurement.

3. Related Work

The evaluation of software is done with the consideration of certain restricting factors. The restricting factors were described by [11] to include: (i) the software user's characteristics which includes experience, age, gender or other features of more specificity, (ii) the role of the user with respect to the types of activities assigned or authorized to perform, (iii) the platform in which the evaluating study is being carried out (such as software laboratory, open-door and unstructured field studies), and (iv) the condition in which the object for evaluation exists. (In this context, the evaluation could be done on a paper prototype, a complementary software, an intermediate software version, or a fully integrated system.) Similarly, a work done by [12] considered web-based evaluation schemes. The work investigated how workflow products with multiple users could be evaluated to understudy their workability and functionality. The work also identified five measurement criteria for evaluating web-based software products to include (i) accuracy, (ii) authority, (iii) objectivity, (iv) currency, and (iv) courage.

The criteria or conditions that qualify a software metric as a tool for measurement were studied by [13]. The study presented an analytic framework which examines a software metric from different perspectives. The study deployed cognitive functional size (CFS) in implementing the analytic framework and succeeded in using the technique to evaluate the complexity of software. The analytic framework, as presented in the study, provided an effective way of representing evaluation parameters which are essential for validating current complexities of software systems. In another development, [14] carried out research with focus on reliability as an important attribute with respect to software quality assurance. The research indicated that stochastic modelling is essential for reliability evaluation. Ref. [15] carried out a study which yielded an evaluation framework for software quality evaluation using multi-criteria decision-making (MCDM) problems. Similarly, [1] carried out a SQuaRE-based study whose objective was to establish a framework that evaluates software quality. The framework's performance was to reduce metric complexities which makes evaluation a less ambiguous process. In the same vein, [16] presented an analytic combination of code complexity metrics for the purpose of fault prediction. This was aimed at enhancing the effectiveness of software quality.

However, the need to accurately perform evaluation on software systems that would present a highly dependable software quality remains a niche for more researches. Following this submission, [8] studied the various software attributes and identified five of them (namely reliability, correctness, maintainability, integrity, and usability) as the most essential for software evaluation. Ref. [8] centered his work on the basis of the attributes which he claimed that

provided valuable indications that the software is in conformation with the specified requirements. The work further performed its selection on the condition that the attribute must have contributed to the software's functional and performance requirements, document development standards, and expected characteristics. In a similar study, [17] identified four quality attributes (namely reliability, efficiency, security and maintainability) as the most relevant in unleashing software quality. The study propounded a couple of guiding principles to control the measurement of attributes which in turn, provides a better knowledge of the performance of a software product and how it can be improved if need be.

4. Metric-based Software Quality Evaluation

The application of software metrics on software attributes is the fundamental activity of software evaluation. Software attributes therefore, are valued by the metrics in numeric terms. Hence, it is of utmost importance to deploy relevant metrics in order to realize a reliable and dependable quality outcome. In this section, six software attributes are identified in adherence to the ISO/IEC 25010 standard [18] and discussed in respect of their metrics which play the role of numeric valuation. This study propounds a model equation for a substantive quality evaluation ($Quality_{eval}$) of software products using the six identified attributes:

$$Quality_{eval} = (\sum_{i=1}^{n} \frac{a_i}{n-1}) * \frac{\ln(n)}{(n-1)^{1.25}} \tag{1}$$

where n = 6, and a_i = attributes (reliability, usability, efficiency, functionality, maintainability and portability). The attributes are derived from the discussions and formulas described in the next session. The application of the six attributes showed that a software is at its best performance when the value of the model tends to unity.

5. Methodology

The six software attributes considered to bring out the peak performance of software systems are elaborated in this section. The corresponding metrics that enable the actualization of their numeric valuation are also identified (as summarized in table 3).

5.1. Reliability

Software reliability is directly concerned with the tendency of correct functioning of a software system via a set of inputs observed within a time frame. As a quality attribute, it brings to limelight, the software's ability to function with little or no interruptions and hence, denotes the probability that the software serves its intended purpose satisfactorily [19,20]. That is to say that reliability finds a way of subduing some inherent errors within the software system. It therefore proves the software's ability to deal with failure, which would occur inevitably [17]. The metrics for measuring software reliability include maturity, fault tolerance and recoverability of the software. The attribute is metrically valued using the function that gets the inverse of the software's mean time to failure (MTTF):

$$R = \frac{1}{MTTF} \tag{2}$$

$$\text{where } MTTF = \sum_{i=1}^{n_e} \frac{t_{i+1} - t_i}{n_e - 1} \tag{3}$$

where n_e = number of failures and t_i = times at which failures occur.

5.2. Usability

Usability of software connotes with the simplicity of the software interfaces which ensures easy usage [21]. Usability is a reflection of the system's user-friendliness and its ability to be comprehended. The metrics for measuring software usability include learnability, memorability, errors and user satisfaction. Usability is ascribed value using the ratio function of the number of successful executions to the number of failures recorded.

$$Usability = \frac{successCount}{failureCount} \tag{4}$$

5.3. Efficiency

Software efficiency is concerned with the system performance. It describes the ability of a system to exhibit the intended functionalities as prescribed in the requirements document. Hence, efficiency points towards easy-to-use condition of a software system. This is a cumulative benefit of a good code structure, software design and the

architectural components (such as databases and web servers) [17]. The metrics for measuring software efficiency include the amount of communications needed among components; functionalities allocated to the components; the way that shared resources are allocated; the algorithm selected for a functionality; and the coding of the selected algorithm [22]. In test-driven development, most developers measure software efficiency using the Root Mean Square Error (RMSE):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\overline{E} - E)^2} \tag{5}$$

where n = number of software executions
$\overline{E}$ = total functions
E = executed functions

In addition, the Mean Magnitude Relative Error (MMRE) is also a useful metrics for evaluating software efficiency. MMRE is given by

$$MMRE(\%) = \frac{1}{n}\sum_{i=1}^{n} MRE * 100 \tag{6}$$

$$\text{where } MRE = \frac{|\overline{E} - E|}{\overline{E}} \tag{7}$$

n = number of projects
$\overline{E}$ = actual effort
E = estimated effort

The advantage of MMRE is that it creates room for comparing software performance over several datasets.

5.4. Functionality

Software functionality lays emphasis on the adherence to the original and approved design of the software. It also ensures that the system is functionally correct as to prove superiority to its rivals. It is also a method that creates room for sizing software products with respect to their functions [23]. The specifications and design architecture of a software system provides the criteria and factors for measuring software functionality. The metrics for measuring software functionality include accuracy, adequacy, interoperability and compliance of software to its intended functions. The software system's function point (FP) is usually a measure for expressing software functionality:

$$FP = CountTotal * CAF \tag{8}$$

where Count-total = Total functions

$$CAF = \left[0.65 + 0.01 * \sum(f_i)\right] \tag{9}$$

$i = 1$ to 14

5.5. Maintainability

Software maintainability is the objective function of software re-engineering. It describes the readiness of software products to be debugged, modified and extended functionally. Maintainability highly concerns software modifiability to ascertain the ease of fixing component errors and/or to adapt to a new environment. A software system is said to be maintainable if it is readable, understandable, and extensible [24]. Hence, they make up the metrics for its measurement. The value or extent of these metrics determines the degree of maintainability possessed by a software product. The software's mean time to repair (MTTR) is usually deployed in test-driven environment to express software maintainability:

$$MTTR = \frac{totalDowntime}{noutages} \tag{10}$$

5.6. Portability

Software portability describes how adaptive a software can function from one execution platform to another. That is, the behavioural stability of software in a different environment. The metrics that validate the measurement of portability include installability, adaptability, replaceability, and compatibility [25]. Test-driven developers often obtain value for portability using the function:

$$Portability = \frac{Number \text{ of successful ports}*100}{Total \text{ number of ports}} \tag{11}$$

Table 3. Attributes and Corresponding metrics

S/N	ATTRIBUTE	CORRESPONDING METRICS
1	Reliability	$R = \frac{1}{MTTF}$
2	Usability	$Usability = \frac{successCount}{failureCount}$
3	Efficiency	$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\overline{E} - E)^2}$
4	Functionality	$FP = CountTotal * CAF$
5	Maintainability	$MTTR = \frac{totalDowntime}{noutages}$
6	Portability	$Portability = \frac{Number \text{ of successful ports}*100}{Total \text{ number of ports}}$

6. Result and Discussion

The metrics identified in table 3 were applied to two software products (depicted as SW1 and SW2). This is aimed at producing values that are required to validate the formulated model equation shown in equation (1). The result of the study showed that the implementation of the metrics with respect to the various attributes gave rise to the output shown in table 4.

Table 4. Output of Quality attributes Valuation

S/N	Attribute	SW1	SW2
1.	Reliability	0.75	0.8
2.	Usability	0.75	0.8
3.	Efficiency	1.5	2.7
4.	Functionality	4.9	7.7
5.	Maintainability	0.25	0.2
6.	Portability	43	36

The output shown in table 4 are graphically represented in fig. 2. The results indicates that both software can be easily maintained owing to their maintainability period of not more than 25 seconds. The results also showed reliability of both software since the tendency to this fact are close to unity (0.75 and 0.80) in both cases. However, SW2 tends to perform more in conformation to its requirements with the output of its functionality at a few units below 10. On the other hand, SW1 proved to be more portable and less prone to failures owing to its less efficiency value. Computing equation (1) using the outputs from the two software, evaluated quality values of 0.93 and 0.86 for SW1 and SW2 respectively were realized. These values are indications that both SW1 and SW2 are of high quality. This is because these values approach unity which is the peak of best performance indication.

7. Conclusion

The prospects of software evaluation with respect to the attributes that can help to reveal software's degree of competence formed the basis of this study. This study is essential considering the pervasive influence of software in the present world in which all facets of life has gotten information technology applications. The study presented software evaluation as a means of delivering reasonable information about the quality of software products. The study made an analytic representation of software evaluation as an aggregate of software metrics and software measurement. To buttress this point, the study presented a relationship model in which metrics and measurement are expressed as inseparable modules of software evaluation where their conjugal interaction yields the quality of software. The study

also presented six identified software attributes whose behaviours and metrics contribute significantly to the revealing of the quality of software systems. The study used this basis to formulate a model equation that can be used to quantify the outcome of software systems evaluation using numeric value. The model equation is operationally active provided the six identified software attributes have been ascribed numeric values using their corresponding metrics. The study further established the fact that measurement and metrics are basic tools for software evaluation and their bonded interactions numerically quantifies the attributes that showcase software quality. Further studies may be carried out to subject metrics to certain factors to determine their relevance towards achieving target outcome. It could also be directed towards creating dynamic evaluation of attributes for optimal valuation of the metrics.

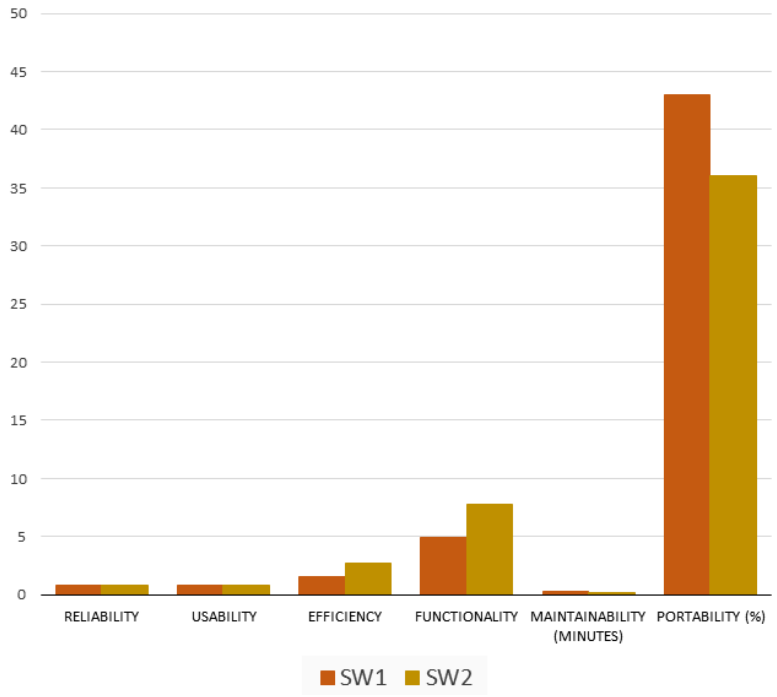

Fig.2. Comparative Representation of Evaluation Output.

References

[1] A. Dewanji, "Software reliability", *Indian Statistical Institute,* Kolkata, pp. 1-26, June 2018, Accessed from http://cs.rkmvu.ac.in/wp-content/uploads/2018/03/Software-Reliability.pdf on 19/07/2022.

[2] A. M. Alashqar, A. A. Elfetouh, and H. M. El-Bakry, "ISO 9126 based software quality evaluation using choquet integral", *International Journal of Software Engineering & Applications (IJSEA)*, vol. 6, no. 1, pp. 111-121, 2015.

[3] A. Zai, R. Butt, and S. Nawaz, "A survey of software quality metrics for software measurement process", *Journal of Software Engineering & Intelligent Systems*, vol. 2, no. 1, April 2017, pp. 49-56.

[4] B. Boehm, "Evaluating Human-Assessed Software Maintainability Metrics", Proceedings from Software Engineering and Methodology for Emerging Domains: 15th National Software Application Conference, NASAC 2016, Kunming, Yunnan, Springer, vol 675, pp. 120, January 2017.

[5] C. Ikerionwu, and I. C. Nwandu, "Quantifying software quality in agile development environment", *Software Engineering*, vol. 9, no. 2, pp. 36-44, 2021.

[6] D. Thakur, "Measuring software quality in software engineering", 2020, Accessed from https://ecomputernotes.com/software-engineering/measuring-software-quality on 26/02/2022.

[7] F. P. Vladicescu, *"Software Reliability Engineering",* Debrecen, 2012, Accessed from https://www.academia.edu/10903371/Software_Reliability_Engineering on 19/07/2022.

[8] Geeksforgeeks, *Software Engineering: Functional Point (FP) Analysis*, 2021, Accessed from https://www.geeksforgeeks.org/software-engineering-functional-point-fp-analysis on 26/02/2022.

[9] G. Gediga, K. Hamborg, and I. Düntsch, "Evaluation of software systems", Institut für Evaluation und Marktanalysen Brinkstr. vol. 19, Germany: 49143 Jeggen, 2015, pp. 1-5.

[10] G. N. Nielsen, *"Introduction to Usability," Current Issues in Web Usability*, 2014, Accessible from http://www.nngroup.com/articles/usability-101-introduction-to-usability on 04/11/2021.

[11] G. O'Regan, *Concise Guide to Software Engineering, in: Fundamentals to Application Methods,* Springer International Publishing, ISBN 978-3-319-57750-0, 2017.

[12] H. Nakai, N. Tsuda, K. Honda, H. Washizaki, and Y. Fukazawa, "A SQuaRE-based software quality evaluation framework and its case study", *IEEE International Conference on Software Quality, Reliability & Security*, pp. 1-4, 2016.

[13] International Organization for Standardization/International Electrotechnical Commission, *Software Engineering - Product quality-part 1: Quality model*, ISO/IEC 25010, 2011.

[14] International Organization for Standardization/International Electrotechnical Commission/ Institute of Electrical and

Electronics Engineers, *Systems and Software Engineering - Vocabulary*, ISO/IEC/IEEE 24765, 2017, pp. 1-522.
[15] I. Sommerville, *Software Engineering*, 10th ed., Pearson Education Limited, ISBN 978-0-13-394303-0, 2016.
[16] K. Stoilova, and T. Stoilov, "Software evaluation approach", *Institute of Computer and Communication Systems*, Bulgarian Academy of Sciences, pp. 1-6, 2005.
[17] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, Addison-Wesley Publishing Company, Inc., Boston, 2003.
[18] N. Fenton, and J. Bieman, *Software metrics: a rigorous and practical approach*, CRC press, 2014.
[19] P. Karuna, M. G. Divya, and N. Mangala, "Statistical analysis of metrics for software quality improvement", *International Journal of Software Engineering & Application (IJSEA)*, vol. 9, no. 1, January 2018, pp. 77-88. DOI: 10.5121/ijsea.2018.9107
[20] P. Karuna, M. G. Divya, C. Sarat, and N. Mangala, "Software quality improvement through statistical analysis on process metrics", *Computer Science & Information Technology (CS & IT)*, CSCP 2017, pp. 39-48. DOI: 10.5121/csit.2017.71804
[21] R. Gunnalan, M. Shereshevsky, and A. Ammar, "Pseudo dynamic metrics [software metrics]", *The 3rd ACS/IEEE International Conference on Computer Systems and Applications*, Cairo, pp. 117-121, 2005.
[22] R. T. Wang, "Reliability evaluation techniques", *Department of Statistics*, Tunghai University, Taichung, Taiwan, pp. 31-69, 2014.
[23] SeaLight, *Measuring Software Quality: A Practical Guide.* 2020, Accessed from https://www.sealights.io/software-quality/measuring-software-quality-a-practical-guide on 04/11/2021.
[24] S. Misra, I. Akman, and R. Colomo-Palacios, "A framework for evaluation and validation of software complexity measures", *Department of Computer Engineering*, Atilim University, Ankara, Turkey, pp. 1-27, 2013.
[25] S. Omri, C. Sinz, and P. Montag, "An enhanced fault prediction model for embedded software based on code churn, complexity metrics, and static analysis results", *Karlsruhe Institute of Technology*, Germany, pp. 1–7, 2019.

Authors' Profiles

Ikenna Caesar Nwandu hails from Imo state, South-East Nigeria. He holds a Masters degree in Computer Science and is currently working on his PhD. His research interests include Software Engineering, Software Processes, Machine Learning and Data Science.

He is an academic staff in the department of Software Engineering, Federal University of Technology Owerri, Nigeria and also an active member of the Nigeria Computer Society (NCS). He is the Departmental Representative at ICT Centre, Federal University of Technology Owerri, Nigeria.
Mr. Nwandu is married with children.

Dr. Juliet N. Odii hails from Imo State Nigeria. She holds a PhD in Computer Science and has risen to the rank of an Associate Professor. She is currently the Acting Head, Department of Computer Science, Federal University of Technology Owerri, Nigeria.

Her research interests are in the areas of Data Structures and Software Engineering. She has a number of publications to her credit.

Dr. Odii is a member of several professional bodies including Nigeria Computer Society (NCS), and National Association of Women in Information Technology (NAWIIT). She is happily married with children.

Dr. Euphemia C. Nwokorie hails from Imo State Nigeria. She is a Doctor of Computer Science and is currently an Associate Professor in the department of Computer Science, Federal University of Technology Owerri, Nigeria. She is the immediate past Head, department of Computer Science, Federal University of Technology Owerri, Nigeria.

She has interest in the areas of Software Engineering, Data Mining, and Operations Research.

Dr. Nwokorie is a Fellow of the Nigeria Computer Society (FNCS), and also belongs to Academia in Information Technology Profession (AITP), and National Association of Women in Information Technology (NAWIIT). She is happily married with children.

Dr. Stanley A. Okolie is a native of Imo State, South-East Nigeria. He holds a PhD in Computer Science and currently works as a Senior Lecturer in the department of Computer Science, Federal University of Technology Owerri, Nigeria. He is the Postgraduate Coordinator in the department.

His research interests include Artificial Intelligence, Engineering & Medicine, Mathematical Computing, and Distributed Computing.

Dr. Okolie is a Fellow of the Nigeria Computer Society (FNCS), and also a National Executive Council Member of NCS, Zonal Coordinator of NCS (South-East), Council Member of Computer Professionals Registration Council of Nigeria. He is married with children.

How to cite this paper: Ikenna Caesar Nwandu, Juliet N. Odii, Euphemia C. Nwokorie, Stanley A. Okolie, "Evaluation of Software Quality in Test-driven Development: A Perspective of Measurement and Metrics", International Journal of Information Technology and Computer Science(IJITCS), Vol.14, No.6, pp.13-22, 2022. DOI:10.5815/ijitcs.2022.06.02