Advances In Multidisciplinary & Scientific Research
A Multidisciplinary & Interdisciplinary Journal

Vol. 2, No. 2, June, 2016

A Benchmarking Method for Traditional Software Quality Metrics

Olaye, E. & Apeh, S.T.
Department of Computer Engineering
University of Benin
Benin City, Nigeria
edoghogho.olaye@uniben.edu; apeh@uniben.edu
+2348061227175

ABSTRACT

The concept and architecture of benchmarking traditional software quality metrics on web-based systems have been demonstrated in the proposed Traditional Software Quality Metrics Benchmark (TSQM Benchmark). The focus of this paper is to extend the architecture of the TSQM Benchmark by presenting a benchmarking method for software quality metrics. The benchmarking method ranks metrics using metrics calculations of web-based systems whose quality level are deliberately and systematically altered by carefully modifying the source code. ISO 9126 quality standard and the web quality model (WQM) were selected as the reference quality models. A web-based system and two traditional software quality metrics were used to demonstrate the ranking ability of the benchmarking method. The benchmark method correctly ranked the selected traditional software metrics in the order of which metric is most suitable for the assessment of specific quality characteristics of a web-based system.

Keywords: Benchmarking, Software Quality Metrics.

Aims Research Journal Reference Format:
Olaye, E. & Apeh, S.T. (2016): A Benchmarking Method for Traditional Software Quality Metrics.
Advances in Multidisciplinary Research Journal. Vol 2, No.2 Pp 81-86.

1. INTRODUCTION

Numerous software quality metrics have been proposed over the years and many tools developed to collect these metrics. These metric tools have been shown to produce different metric values for the same software system (Jain, Srivastava, and Katiyar, 2014). Many theoretical and empirical validation methodologies for software metrics have been extensively studied in literature (Srinivasan and Devi, 2014) but metrics selection is mostly arbitrary for developers of web-based systems. Gracia-Castro argues that benchmarking offers more benefits than evaluation through continuous improvement and recommendations on best practices (García-Castro, 2008). Sim *et al*., (2003) advocated for the definition of benchmarks in software engineering areas. However, even though very few benchmarks studies have been conducted, there is no established method to determine the degree (in quantitative terms) in which a given metrics is better suited than other metrics for quality measurement of a web-based system. Demyanova *et al*., (2006) suggested the use of empirical software metrics for benchmarking verification tools (Demyanova, Pani, Veith, and Zuleger, 2015). Blackburn, *et al*., (2006) developed the DaCapo Benchmarks to foster innovation in system design and implementation for Java and other managed languages (Blackburn*, et al*., 2006). Rentrop, (2006) investigated the uses of software metrics as benchmarks for source code (Rentrop, 2006).

Other benchmarking studies not related to software metrics include YCSB Client (Cooper, Silberstein, Tam, Ramakrishnan, and Sears, 2010), the SPEC Web 2009 benchmark (SPEC, 2009), benchmark optimization software with performance profiles (Dolan and Moré, 2002), benchmarking software development activity (Maxwell and Forselius, 2000), and the Michigan benchmark (Runapongsa, Patel, Jagadish, and Al-Khalifa, 2002). Olaye and Apeh, (2016) proposed a Traditional Software Quality Metric Benchmark (TSQM Benchmark). The benchmark was designed to establish a standard basis for selecting and applying traditional software metrics on web-based systems. The concept and architecture for the TSQM benchmark research is reported elsewhere (Olaye and Apeh, 2016). This paper presents an extended version of the TSQM architecture and reports on the specific benchmarking method used in the TSQM Benchmark.

2. METHODOLOGY

The TSQM Benchmarking tool architecture which is comprised of components such as MASU (Metrics Assessment plugin platform for Software Unit) (Higo, *et al.*, 2011), and QMetrics (Schackmann, Jansen, Lischkowitz, and Lichter, 2009) is shown in Figure 1. Though the tool is still under development, its architecture was the framework used to develop the benchmarking method. The Benchmarking Unit implements the benchmarking method to rank traditional software quality metrics. The benchmarking method focuses on quality characteristics defined in the ISO/IES 9126 Quality model and the Web Quality Model (WQM). There are other models such as SQuaRE(Azuma, 2001) but ISO/IEC 9126 was selected because it has been extensively studied (Jung, Kim, and Chung, 2004), and WQM model was added because it was successfully used to categorize more than three hundred web metrics (Calero, Ruiz, and Piattini, 2005). A metric tool called "locMetrics" ("locmetrics,") was used as a substitute to collect metrics values because the plugin management unit of the TSQM Benchmarking tool is still under development. LocMetrics collects metrics ten different metrics such as Lines of Code (LOC), McCabe v(G) complexity and Comments Lines (CLOC). A single web-based system called "GoalsJournal" was provided to the system. GoalsJournal is an open source web-based application written mainly in the java language for tracking daily habit. The Source code pre-processing unit of the TSQM Benchmark tool was used to separate the program code from the other software artefacts. The source code of the selected web-based system was examined in Notepad++ and Eclipse IDE was used to modify the source codes to alter the quality. Correlation analysis was performed using MATLAB software.

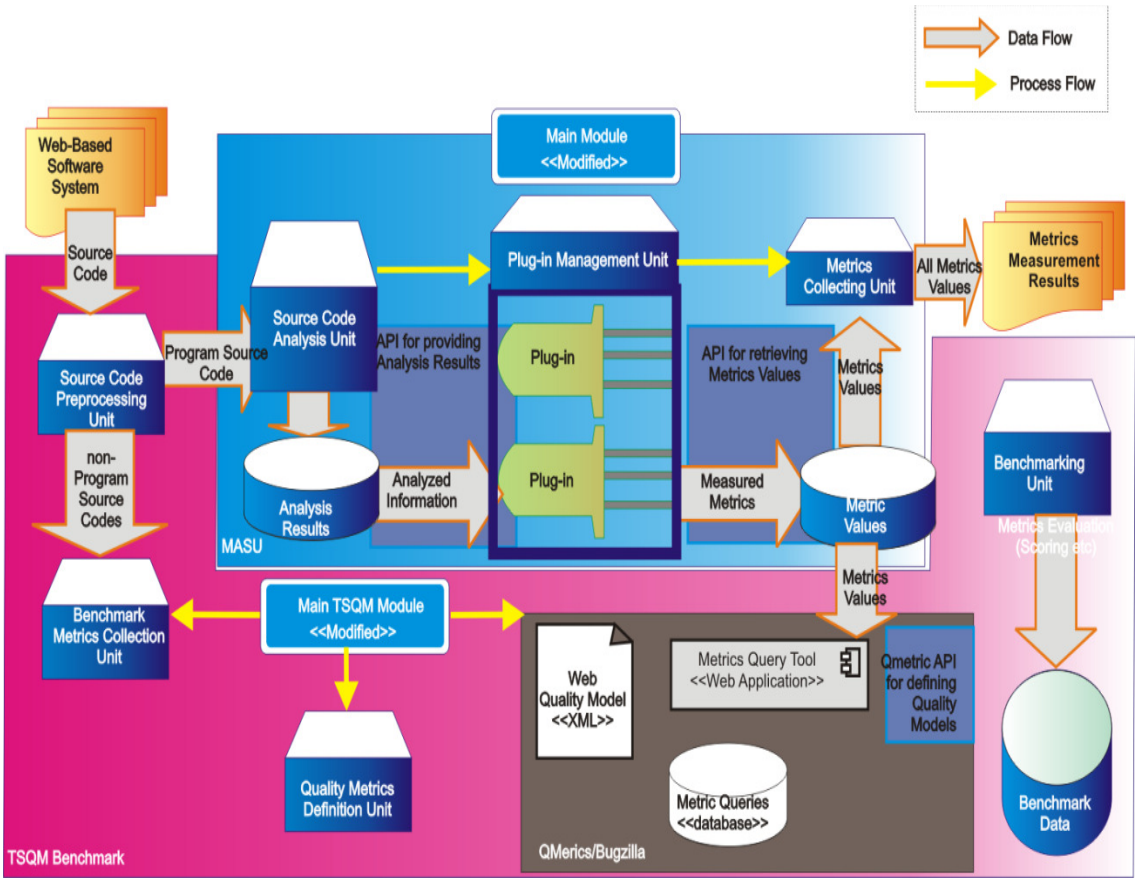

Figure 1: TSQM Benchmarking Tool Architecture (Olaye and Apeh, 2016)

2.1 The TSQM-Benchmarking Method

The goal of the benchmarking method is to generate empirical data to rank traditional software quality metrics based on the degree of suitability of measuring a specific software quality attribute. The ranking is based on answers to the following questions related to statistical analysis of experimental results:

A. Which metrics behave in the most consistent manner?
B. To what extent does a metric M_i behave in a more consistent and logical mode than metric M_j in quantifying the quality of a web-based system?
C. To what extent does the value for metric M_i on a given web-based system change when the quality of the web-based system is improved or worsened?

The flowchart for the TSQM Benchmark method is shown in Figure 2. The first step of the method is to calculate traditional software metrics values for selected web-based systems. The web-based system is then deliberately and scientifically altered to reduce the level of each of its quality attribute as defined in the ISO quality model (ISO/IEC standard 9126) and the Web Quality model. The metrics is then recalculated to generate new metric values. This process is repeated to generate metric values for corresponding quality levels. The metrics ranking is achieved by analysing the metric values.

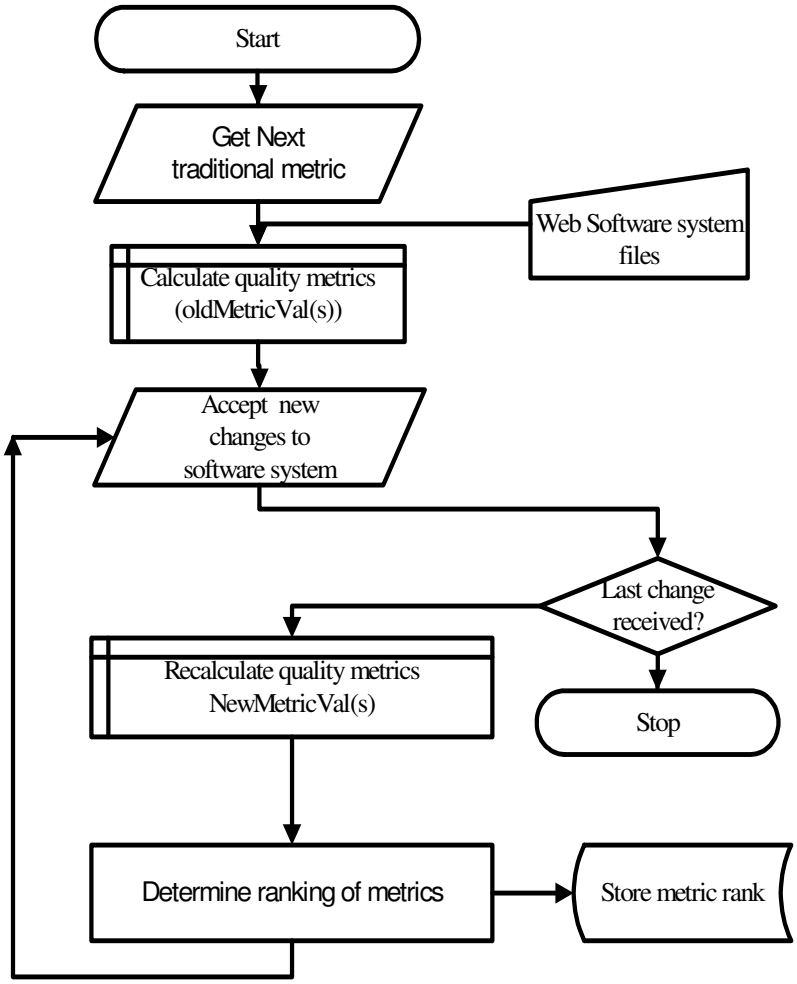

Figure 2: Flowchart for the TSQM Benchmark

2.2 Experimental Analysis of the benchmarking method

The benchmarking method was analysed using the experimental illustrating presented in Table 1. For simplicity, only two internal quality characteristic of the ISO/IEC 9126 model is considered; (1) Usability (understandability, operability, learnability, attractiveness, and usability compliance) and (2) Maintainability (analysability, changeability, stability, and maintainability compliance) are represented in the analysis.

Table 1: Benchmarking Activities

Benchmarking Activity	Quality Attribute (ISO/IEC 9126)	Modification to web-based software system	Metric Value (LOC)	Metric Value (v(G))
		No modification (QL0)	M0	CC0
1	Operability	Reduce quality by removing all buttons (QL1)	M1	CC1
2	Understandability	Reduce the quality by removing all user visible text (QL2)	M2	CC2
3	Learnability	Remove label for each textbox (QL3)	M3	CC3
4	Attractiveness	Reduce quality by inverting colour (QL4)	M4	CC4
5	Usability Compliance	Reduce quality by removing implementation of a commands (QL5)	M5	CC5
...	…	...		
11	Analysability	Reduce quality by introducing nested loops (QL11)	M5	CC6
12	Analysability	Reduce quality by changing loop variables and structure (QL12)	M6	CC6

For each benchmarking activity, the web-based system is modified so that the quality attribute is altered. For example, Activity 3 (remove label from each textbox) could be achieved by modifying the source code as follows:

Original Source Code: jTextPane3.setText(recursosTexto.getString("TomorrowTPTxt"));

ModifiedSource Code: jTextPane3.setText(recursosTexto.getString(""));

In Table 1, it is expected that if LOC is a good measure of usability the following condition should be true: $M_i < M_0 \ (for\ i = 1, 2, 3, 4 \ldots n)$. Where n is the number of benchmark activities. Similarly, if v(G) is a good measure of useability, the condition $CC_i < CC_0 \ (for\ i = 1, 2, 3, 4 \ldots n)$ must be true. In addition to the simple deductions, the Spearman's rank coefficient given by Equation 1 is used for correlation analysis.

$$R_s = 1 \quad \frac{6 \sum d^2}{n(n^2 - 1)} \qquad (1)$$

Where d is the difference between the two ranks of each observation and n is the number of observations. For a given metrics, positive or negative correlation means that the metric is a good measure of the specific quality. Positive correlation applies to metrics that gives higher values for better quality while negative correlation applies to metrics that give lower values for lower quality. Since the LOC metrics follows the later rule, it is expected to have positive correlation with the established quality ranking.

3. RESULT PRESENTATION AND DISCUSSIONS

Some of the results of the illustration of the benchmarking method using GoalJournal web-based system is presented in Table 2. Only 10 activities out of the 20 activities in the experiment are shown, other aspects of the table contains many repeated values which are excluded to reduce space occupied by the table in this paper.

Table 2: Some results of benchmarking method illustration

Benchmarking Activity	Quality Attribute (ISO/IEC 9126)	Modification to web-based software system	Metric Value (LOC)	Metric Value (vG)
		No modification	5335	447
1	Operability	Reduce quality by removing all buttons (QL1)	5309	447
2	Understandability	Reduce the quality by removing all user visible text (QL2)	5309	447
3	Learnability	Remove label for each textbox (QL3)	5309	447
4	Attractiveness	Reduce quality by inverting colour (QL4)	5309	447
5	Usability Compliance	Reduce quality by removing implementation of commands (QL5)	5309	447
11	Analysability	Reduce quality by introducing nested loops (QL11)	5358	451
12	Analysability	Reduce quality by changing loop variables and structure (QL12)	5361	462
13	Analysability	Reduce quality by changing loop variables and structure (QL13)	5361	462
14	Analysability	Reduce quality by adding complex branch structures (QL14)	5392	479
15	Analysability	Reduce quality by changing loop variables and structure (QL15)	5392	479

Using results from twenty benchmarking activities a rank coefficient of -0.5343 and 0.6395 was obtained for the LOC metrics and v(G) respectively. This results show that there is less negative correlation for LOC and more positive correlation for v(G). However, considering the userbility characteristics alone, there is no significant correlation between the actual quality levels (QL1 to QL10) and all the corresponding metrics values for the first 10 benchmarking activities. The implies that LOC and v(G) are not good measures of usability. It is common knowledge that LOC and v(G) metrics are not good measure of usability and the result obtained using the benchmarking method agrees with this fact. On the other hand, the illustration shows that v(G) can significantly measure analysability better that LOC. Again this agrees with common knowledge because LOC is a size metrics while v(G) is a complexity metrics. These deductions are summarised in Table 3. In Table 3(a) the rankings was not achieved because none of the metrics are suitable for measuring usability. Table 3(b) shows that v(G) ranks higher than LOC in the measurement of maintainability characteristics.

Table 3: (a) Usability Ranking for LOC and v(G) metrics

Rank	Metric	Comments
x	LOC	Not Suitable for usability
x	V(G)	Not Suitable for usability

Table 3 (b) Maintainability Ranking for LOC and v(G) metrics

Rank	Metric	Comments
1	V(G)	Suitable for maintainability
2	LOC	Suitable for maintainability

A major limitation of the benchmarking method described in this paper is the difficulty in automating the quality reduction process for web-based systems used in the benchmarking process. The method involves direct modification of source code of the web-based system by the experimenter thus introducing some level of subjectivity. This limitation is mitigated by ensuring that such modification is based on the quality characteristics specified in a quality standard and not the metrics definitions. Secondly, ensuring that the modification is intentionally done to reduce the quality of an existing web-based system offers the advantage of reducing bias on the part of the experimenter.

4. CONCLUDING REMARKS

This paper presented and demonstrated a benchmarking method for traditional software quality metrics. The benchmarking method was illustrated using two internal quality characteristic of the ISO/IEC 9126 model and it was used to rank the LOC and (G) metrics. The paper has illustrated shows that the benchmarking method is fair and produces accurate rankings.

REFERENCES

1. Azuma, M. (2001). *SQuaRE: the next generation of the ISO/IEC 9126 and 14598 international standards series on software product quality.* Paper presented at the ESCOM (European Software Control and Metrics conference).
2. Blackburn, S. M., Garner, R., Hoffmann, C., Khang, A. M., McKinley, K. S., Bentzur, R.*, et al.* (2006). *The DaCapo benchmarks: Java benchmarking development and analysis.* Paper presented at the ACM Sigplan Notices.
3. Calero, C., Ruiz, J., and Piattini, M. (2005). Classifying web metrics using the web quality model. *Online Information Review, 29*(3), 227-248.
4. Cooper, B. F., Silberstein, A., Tam, E., Ramakrishnan, R., and Sears, R. (2010). *Benchmarking cloud serving systems with YCSB.* Paper presented at the Proceedings of the 1st ACM symposium on Cloud computing.
5. Demyanova, Y., Pani, T., Veith, H., and Zuleger, F. (2015). *Empirical Software Metrics for Benchmarking of Verification Tools.* Paper presented at the Computer Aided Verification.
6. Dolan, E. D., and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical programming, 91*(2), 201-213.
7. García-Castro, R. (2008). *Benchmarking Semantic Web Technology.* Informatica.
8. Higo, Y., Saitoh, A., Yamada, G., Miyake, T., Kusumoto, S., and Inoue, K. (2011). *A pluggable tool for measuring software metrics from source code.* Paper presented at the Software Measurement, 2011 Joint Conference of the 21st Int'l Workshop on and 6th Int'l Conference on Software Process and Product Measurement (IWSM-MENSURA).
9. Jain, S., Srivastava, V., and Katiyar, P. (2014). Integration of Metric Tools for Software Testing. *International Journal of Enhanced Research in Science Technology & Engineering, ISSN*, 2319-7463.
10. Jung, H.-W., Kim, S.-G., and Chung, C.-S. (2004). Measuring software product quality: A survey of ISO/IEC 9126. *IEEE software, 21*(5), 88-92.
11. locmetrics. from www.locmetrics.com
12. Maxwell, K. D., and Forselius, P. (2000). Benchmarking software development productivity. *Software, IEEE, 17*(1), 80-88.
13. Olaye, E., and Apeh, S. T. (2016). *Towards Benchmarking of Traditional Software Quality Metrics on Web-based Systems.* Paper presented at the 6th iSTEAMS Multidisciplinary Cross-Border Conference, University of Professional Studies, Accra Ghana
14. Rentrop, J. (2006). *Software Metrics as Benchmarks for Source Code Quality of Software Systems.* Universiteit van Amsterdam.
15. Runapongsa, K., Patel, J. M., Jagadish, H., and Al-Khalifa, S. (2002). *The Michigan Benchmark: A microbenchmark for XML query processing systems.* Paper presented at the EEXTT.
16. Schackmann, H., Jansen, M., Lischkowitz, C., and Lichter, H. (2009). *QMetric-a metric tool suite for the evaluation of software process data.* Paper presented at the ICSE Companion.
17. Sim, S. E., Easterbrook, S., and Holt, R. C. (2003). *Using benchmarking to advance research: A challenge to software engineering.* Paper presented at the Proceedings of the 25th International Conference on Software Engineering.
18. SPEC. (2009). SPEC Web 2009. from Standard Performance Evaluation Corporation: https://www.spec.org/web2009/
19. Srinivasan, K., and Devi, T. (2014). Software Metrics Validation Methodologies in Software Engineering. *International Journal of Software Engineering & Applications, 5*(6), 87.