DOI: http://dx.doi.org/10.26483/ijarcs.v8i8.4835

ISSN No. 0976-5697

Volume 8, No. 8, September-October 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

AUTOCORRELATION WEIGHTED SUM ENTROPY BASED SOFTWARE QUALITY MANAGEMENT FOR OPEN SOURCE APPLICATION

R.Chennappan
Research Scholar,
Department of Computer Science
Periyar University
Salem, Tamilnadu,India

Dr. Vidyaa Thulasiraman
Head & Assistant Professor
Department of Computer Science
Govt. Arts & Science College for Women,
Bargur,Krishnagiri, Tamilnadu, India

Abstract : Software metrics is used to evaluate software systems quality and to improve the software reliability. Recently, few researches have been developed for enhancing the quality of open source software using Software metrics. But, the software quality management performance of existing works was not efficient while performing multiple software operations which affect the software reliability. To attain higher scalability rate with reduced service provisioning time while improving the reliability of software quality, a component model called Autocorrelation Weighted Sum Entropy (AWSE) technique is proposed. To minimize software quality degradation while performing multiple software operations, a service provisioning time entropy is considered. The AWSE technique initially measures autocorrelation function for consecutive versions of same application to find the relationship between a conventional versions and contemporary version. After that, AWSE technique computes autocorrelation for service provisioning time entropy that considers the effect of maintenance operations carried out both on contemporary versions and on the conventional versions. Then, AWSE technique measures average time entropy to obtain the time entropy of consecutive versions of same application. This in turn helps for reducing the service provisioning time and improving the scalability of software quality management. Finally, AWSE technique used Weighted Sum Entropy (WSE) model that considers the Mean Time between Failure (MTF) to improve the software reliability in a specified environment for a given amount of time and to reduce the cost of software quality testing. The AWSE technique conducts the experimental works on parameters such as scalability, service provisioning time and software reliability. The experimental result shows that the AWSE technique is able to improve the scalability and also reduces the service provisioning time of software quality management when compared to state-of-the-art-works.

Keywords: software quality, open source software, Autocorrelation, software quality management, Weighted Sum Entropy

1. INTRODUCTION

Software quality plays a significant importance in software development projects because it affects the every aspect of the system such as the functionality, reliability, availability, maintainability, and safety. Software development process requires information about almost all aspect of the software development phase such as objectives, monitoring and control of activities, project costs and technical quality. Besides, software metrics is employed for evaluating software systems quality and improving the software reliability.

Recently, few researches have been developed for increasing the quality of software. A systematic framework was intended in [1] by using Markov chain for modeling the stochastic processes of a quality management system and selection of the optimum set of factors impacting software quality. Systematic framework improves the software reliability. But, scalability of software quality management was remained unaddressed. Iterative redundancy method was intended in [2] for improving the software system reliability. But, reducing the cost of software quality testing was remained unsolved.

An initial design and development of an integrated analyser component was intended in [3] for enlarging the functionality of the open source framework for software quality management. But, multiple software maintenance handled over time results in software quality degradation and also increases the service provisioning time. A novel method was designed in [4] to examine the relationship between quality assurance and software ecosystems in which set of quality attributes are used to ensure software quality assurance. However, software reliability was remained unaddressed.

In [5], reliability metrics are employed for quantitative measurement of software reliability and to evaluate of reliability of open source software. But, the quantitative estimation is not sufficient for software quality management. A systematic literature review of open source software quality assessment models was presented in [6] to find out the important quality attributes with which to develop more reliable quality models. However, open source software quality evaluation across different domains was remained unsolved.

In [7], the evolution of Mozilla Firefox from a traditional release model to a rapid release model was analyzed to determine potential changes in field quality (users) and bug fixing (developers). A novel approach was designed in [8] for evaluating the stability of open-source software systems with aid of combination of Bayesian Classifiers.

In [9], the quality evolution of an open source Java software system was examined with aid of metrics in which software quality was addressed from an internal point of view. However, the number of defects as a quality indicator was considered. A novel method was designed in [10] for measuring reliability of an open source software using computational systems through considering both hardware and software failure impacts. But, scalability was remained unsolved. To overcome the above mentioned existing issues, an Autocorrelation Weighted Sum Entropy

(AWSE) is developed. The research objective of AWSE technique is formulated as follows,

- To enhance the scalability and to reduce the service provisioning time of software quality management, autocorrelation function is used in AWSE technique.
- To improve the software reliability and to reducing the total cost during the software quality testing, Weighted Sum Entropy model is employed in AWSE technique.

The rest of this paper is organized as follows. Section 2 explains an Autocorrelation Weighted Sum Entropy (AWSE) based software quality management with the assist of architecture diagram. Section 3 and Section 4 explains the experimental settings and details performance analysis with the aid of parameters. Section 5 describes the related works. Finally, Section 6 concludes this paper.

2. AUTOCORRELATION WEIGHTED SUM ENTROPY TECHNIQUE

One of the significant objectives of the software engineering is to design techniques for high-quality software solutions. Software managers and developers employ software metrics to measure and enhance the quality of a software solution during the development process. These metrics evaluate the quality of open source software attributes like product size and complexity. Besides, Software quality management is a process which manages the quality of open source software to ensure the product which satisfies the user needs. The objective of software quality management is to check the product meets the quality standards expected by the user.

Recently, many researches works has been designed for improving the reliability of open source software. However, the software quality management performance was not sufficient while performing multiple software operations which lack the reliability of software. In order to improve the software quality management performance with higher reliability rate and reducing the total cost involved during software quality testing, an Autocorrelation Weighted Sum Entropy (AWSE) technique is introduced. The overall architecture diagram of AWSE technique for software quality management is shown in below Figure 1.

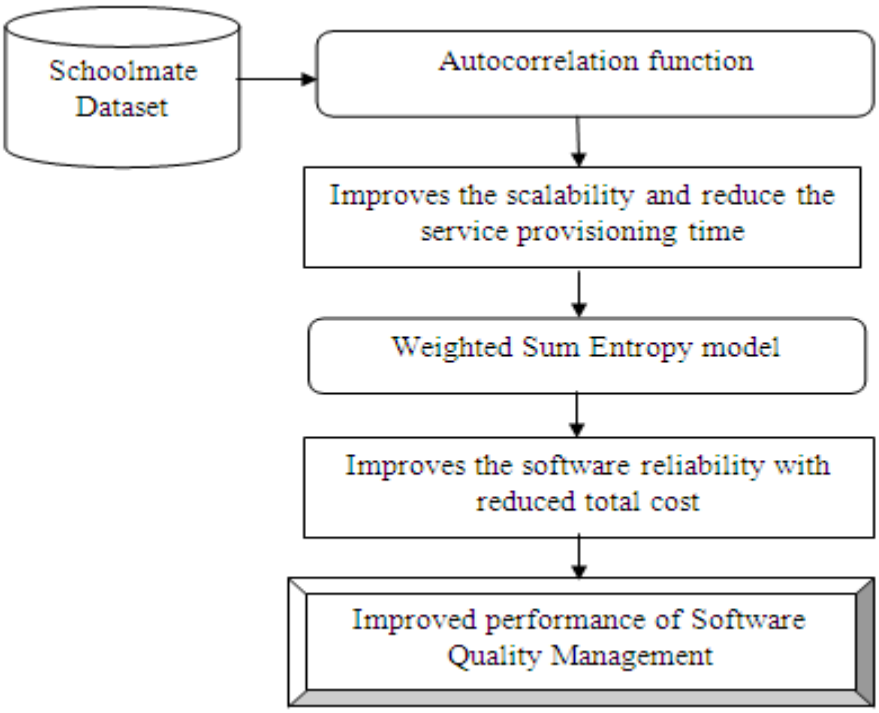

Figure 1 Process of Autocorrelation Weighted Sum Entropy Based Software Quality Management

As shown in Figure 1, AWSE technique takes schoolmate dataset as input. Then, AWSE technique used Autocorrelation function with objective of increasing the scalability and minimizing the service provisioning time of software quality management. After that, AWSE technique employed Weighted Sum Entropy model in order to improve the software reliability and reducing the total cost of software quality testing. As a result, AWSE technique improved performance of software quality management. The detailed explanation about AWSE technique is described in following sections.

2.1 Autocorrelation Function

In proposed technique, Autocorrelation Function is used to measures the relationship between a contemporary version and the conventional versions of same open source application. The resulting output of autocorrelation function is ranges between $+1$ to -1. An autocorrelation of 1 signify a perfect positive correlation. On the other hand, an autocorrelation of negative 1 denotes perfect negative correlation. Thus, autocorrelation function over a time is measured for consecutive versions of same open source application to improve the software reliability. The process of autocorrelation function for improving the scalability and

reducing the service provisioning time of software quality management is shown in below Figure 2.

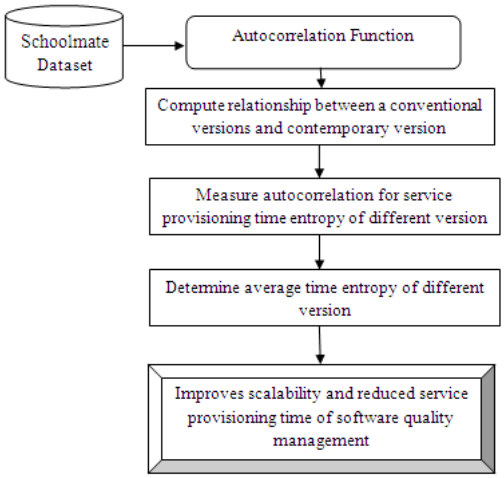

Figure 2 Process of Autocorrelation Function for Software Quality Management

As shown in Figure 2, the autocorrelation function initially measures the correlation between contemporary version Y_t and the conventional versions Y_s that are both part of the same open source application. Besides autocorrelation between Y_t and $Y_s$based on the differentiation or lag between t and s.Thus $\tau = t - s$. Therefore, the autocorrelation ρ_τ function is defined as the correlation between the versions of open source software application separated by lag τ which is mathematically formulated as below,

$$\rho_\tau = \frac{c_\tau}{c_0} \quad (1)$$

From the equation (1), c_τ represents the sample auto covariance function whereas c_0 denotes the variance of stochastic process. The sample auto covariance function is mathematically expressed as,

$$c_\tau = \sum_t^{N-k} \frac{(Y_t - \bar{Y})(Y_{t+\tau} - \bar{Y})}{N} \quad (2)$$

Thus, for a given different versions of same open source software application $Y_1, Y_2, Y_3, \ldots, Y_N$ at time $t_1, t_2, t_3, \ldots, t_N$, the autocorrelation function is defined as follows,

$$\rho_\tau = \frac{\sum_{i=1}^{N-k}(Y_i - \bar{Y})(Y_{i+k} - \bar{Y})}{\sum_{i=1}^{N}(Y_i - \bar{Y})^2} \quad (3)$$

From the equation (3), autocorrelation function for consecutive versions of open source application is determined. This in turn provides the differentiation between the versions of same open source software program to enhance the software quality.

The determined auto correlation is depends only on the lag between t and s where t denotes contemporary version and s refers the conventional versions of open source software program. Therefore, the lag $\tau = t - s$ indicates a period of time between contemporary version and conventional versions of same open source software. Thus, the autocorrelation function of time lag is mathematically expressed as,

$$R(\tau) = \frac{[(Y_t - \mu)(Y_{t+\tau} - \mu)}{\sigma^2} \quad (4)$$

From the equation (4), μ denotes mean and σ^2 indicates variance of time. In addition, autocorrelation for service provisioning time entropy is evaluated that considers the effect of maintenance operations carried out for both contemporary versions and the conventional versions. Hence, the autocorrelation for service provisioning time entropy computes the amount of time taken for creating the contemporary version of open source application with the conventional versions. Thus, the autocorrelation for service provisioning time entropy of different versions is measured using following mathematical expression,

$$\rho_\tau[SPTE] = \frac{E[(Y_t - \mu)(Y_{t+\tau} - \mu)]^2}{E[(Y_t - \mu)^2]} \quad (5)$$

From the equation (5), the autocorrelation for service provisioning time entropy is estimated in which E represents the entropy that denotes the maximum time at which the conventional versions of open source software is updated into the contemporary version. Then the modern version of open source software is provided to the user. Furthermore, Average time entropy is estimated for consecutive versions of same open source software application by using following mathematical representation,

$$\rho_\tau[ATE] = \frac{1}{N}\sum_{i=0}^{N} \rho_\tau[SPTE]_i \quad (6)$$

From the equation (6), average time entropy is arrived at with which the time entropy of different versions is obtained for consecutive versions of same open source software application, reducing the service provisioning time and improving the scalability. The scalability is measured in terms of number of versions is adoptable for a given open source software program to increase the open source software quality or reliability. The algorithmic process autocorrelation function for software quality management to improve the software quality is shown in below algorithm 1.

Input: schoolmate dataset
Output: Improved Scalability with Reduced Service provisioning time
Step 1: Begin
Step 2: **For** each open source software program
Step 3: Measure auto correlation function for consecutive versions of same application using (3)
Step 4: Compute autocorrelation function of time lag using (4)

Step 5: Measure autocorrelation for service provisioning time entropy using (5)
Step 6: Compute Average time entropy using (6)
Step 7: **End for**
Step 8:End

Algorithm 1 Autocorrelation Function Based Software Quality Management

By using the above algorithmic process, AWSE techniques initially determines the correlation among the different versions for each open source software program. After that, AWSE techniques estimate a period of time between contemporary version and conventional versions of open source software. Subsequently, AWSE techniques computes service provisioning time entropy to find out the amount of time necessitated for constructing the contemporary version of open source application with the conventional versions. Finally, AWSE techniques calculate the average time entropy to get the entropy of consecutive versions of same open source software application. As a result, an AWSE technique increases the scalability and reduces the service provisioning time of software quality management in an effective manner.

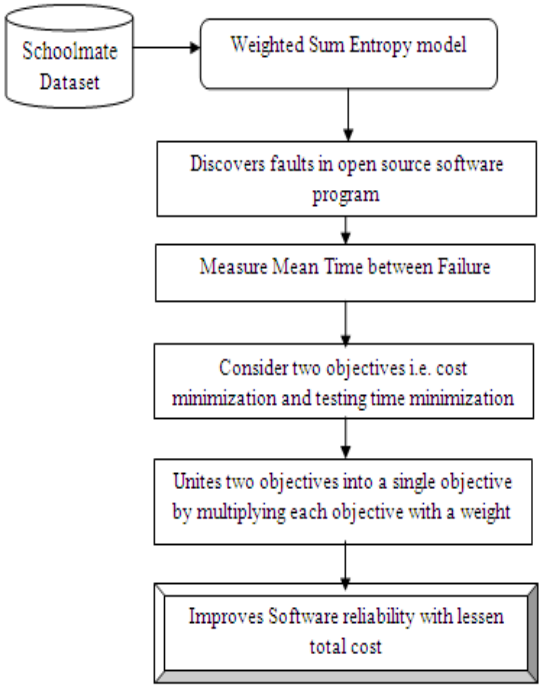

Figure 3 Weighted Sum Entropy Model for Software Reliability

Figure 3 shows the block diagram of Weighted Sum Entropy model for improving the reliability of open source software program during software quality management. Software reliability determines the probability that open source software will work properly in a specified environment and for a given amount of time. Software reliability increases when the faults are removed from the program. By using the following mathematical formula, the mean time between failures (MTF) is measured that find outs the probability of failures in given open source software.

$$MTF = Mean\ Time\ To\ Failure\ (MTTF) + Mean\ Time\ To\ Repair\ (MTTR) \quad (7)$$

2.2 Weighted Sum Entropy model In AWSE techniques, a Weighted Sum Entropy model is used to reduce the total cost involved along with improving the reliability for open source software program in a given amount of time during software quality testing. The Weighted Sum Entropy model considers the Mean Time between Failure (MTF) to enhance the software reliability in a specified environment for a given amount of time. MTF measures the reliability of given open source software to improve its quality and also helps to identify many defects in the software design and functionality. Besides, Weighted Sum Entropy model considers the two objectives namely cost minimization and testing time minimization with objective of reducing the total cost involved while improving the reliability of software quality. The process of Weighted Sum Entropy model for improving the software reliability is shown in below Figure 3.

From the equation (7), MTTF denotes the differentiation of time between two consecutive failures and MTTR represents the time required to resolve the failure in open source software. Besides, the probability of failures i.e. number of defects in given open source software program is identified by using following equation,

$$Failures_P = \frac{Number\ of\ failing\ test\ cases}{Total\ number\ of\ cases\ under\ consideration} \quad (8)$$

From the equation (8), number of defects in given open source software is identified. After identifying the number of faults in open source software, The WSE model considers the two objectives namely cost minimization and testing time minimization in order to reduce the total cost involved while improving the reliability of software quality. These two objectives is mathematically formulated as,

$$F(x) = [F_1(x), F_2(x)] \quad (9)$$

From the equation (9), $F_1(x)$ denotes cost required for improving the software quality whereas $F_2(x)$ refers the time required for testing the overall open source software system. The cost for improving the software quality measures the cost incurred while sharing information between users that include source code, test cases and operational knowledge. The cost is measured in terms of amount of memory utilized for storing the source code, test cases and operational knowledge. Thus, cost of software quality improvement is mathematically formulated as,

$$cost = SC_M + TC_M + OK_M \quad (10)$$

From the equation (10), SC_M indicates the amount of memory utilized for storing the given open source software code and TC_M denotes the memory consumed for

test cases to test the software reliability whereas OK_M refers the memory taken for storing the operational knowledge of software system. Second objective is reducing the time for overall test system infrastructure that include the test cases testing time and other test related information between users which is measured by using following formula,

$$testing\ time = n * time(testing\ one\ testcase\ )$$
(11)

From the equation (11), the amount of time needed for testing the overall open source software system is determined in which n indicates the number of test cases used. Thus, the Weighted Sum Entropy model for two objectives i.e. cost minimization and testing time minimization is mathematically represented as,

$$U = \sum_{i=1}^{k} w_i F_i(x)$$
(12)

From the equation (12), w_i are weights $i = 1,2,3,..,k$ corresponding to objective functions which satisfy the following conditions,

$$\sum_{i=1}^{k} w_i = 1, w_i \geq 0, i = 1,2,3,..,k$$
(13)

The Weighted Sum entropy Model is widely used for multi-objective optimization problems. It integrates the diverse objectives and weights corresponding to those objectives to construct a single objective for each alternative to make them comparable. Therefore, Weighted Sum entropy Model combines two objectives i.e., cost minimization and time for test system infrastructure into a single objective by multiplying each objective with a weight to lessen the total cost involved while improving the reliability of software quality which is formulated as,

$$U = w_1 F_1(x) + w_1 F_2(x)$$
(14)

From the equation (14), w_1 and w_2 is the assigned weights whereas $F_1(x)$ indicates cost required for improving the software quality whereas $F_2(x)$ represents the time required for testing the overall software system. The variable w_1 and w_2 are related based on the following expression,

$$w_2 = 1 - w_1$$
(15)

By using these weights, the significance of the objective function can be tuned according to system requirement to enhance its reliability. Therefore, WSE model reduces the total cost involved along with improving the reliability for a given amount of time during software quality testing. The algorithmic process of Weighted Sum Entropy model for improving the reliability of open source software program during the software quality management is shown in below algorithm 2.

Input: schoolmate dataset
Output: Improved software reliability with reduced total cost
Step 1: Begin
Step 2: **For** each open source software program
Step 3: Find number of faults using (8)
Step 4: Measure mean time between failures using (7)
Step 5: Compute cost required for improving the software quality using (10)
Step 6: Evaluate amount time required for testing the given open source software program using (11)
Step 7: Combines two objectives into a single objectives to reduce the total cost involved during software quality testing using (14)
Step 7: **End for**
Step 8: End

Algorithm 2 Weighted Sum Entropy model

By using the above algorithmic process, AWSE technique initially identifies the number of faults in each open source software program. Then, an AWSE technique computes the mean time between failures to find out the amount of time required for finding and resolving the faults in given open source software program. Subsequently, AWSE techniques calculates the cost required and amount of testing time required for improving the software quality. Finally, AWSE technique combines two objectives into a single objective to reduce the total cost involved during software quality testing. This in turn helps for improving the reliability of open source software program.

3. EXPERIMENTAL SETTINGS

The proposed Autocorrelation Weighted Sum Entropy (AWSE) technique is implemented in Java Language using schoolmate data set. The AWSE technique used schoolmate data set which includes of numerous PHP open source software program for improving reliability of software quality. The performance of AWSE technique is measured in terms of scalability, service provisioning time and software reliability.

4. RESULT AND DISCUSSIONS

In this section, the result analysis of AWSE technique is estimated. The effectiveness of AWSE technique is compared against with two methods namely systematic framework [1] and Iterative redundancy method [2] respectively. The efficiency of AWSE technique is evaluated along with the following metrics with the help of tables and graphs.

4.1 Measurement of Scalability

The scalability measures the capability of AWSE technique to handle a huge size of open source software program for improving the reliability of software. The scalability is

measured in terms of percentage (%). While the scalability is higher, the method is said to be more efficient.

Table 1 Tabulation for Scalability

Size of software program code (KB)	Scalability (%)		
	Systematic Framework	Iterative Redundancy Method	AWSE technique
10	63.18	71.56	80.23
20	63.98	74.65	81.95
30	65.14	75.88	83.26
40	67.85	77.36	85.92
50	70.26	79.52	88.16
60	71.82	80.80	89.90
70	73.64	81.64	91.03
80	75.99	82.95	91.89
90	76.59	83.65	92.65
100	78.23	85.84	94.42

Table 1 demonstrates the result is obtained for scalability with respect to different size of software code in the range of 10-100 KB using three methods. The AWSE technique considers the framework with diverse size of open source software code for improving the reliability of software quality. While 50 KB open source software code is taken for software quality management, proposed AWSE technique achieves the 88.16 % scalability whereas the systematic framework [1] and Iterative redundancy method [2] achieves the 70.26% and 79.52% respectively. autocorrelation function in proposed AWSE technique. With aid of autocorrelation function, proposed AWSE technique efficiently discovers the relationship between the contemporary version and conventional versions of same open source software. This in turn helps for updating the conventional versions. As a result, the proposed AWSE technique improves the scalability of software quality management by 25% when compared to systematic framework [1] and 11% when compared to Iterative redundancy method [2] respectively.

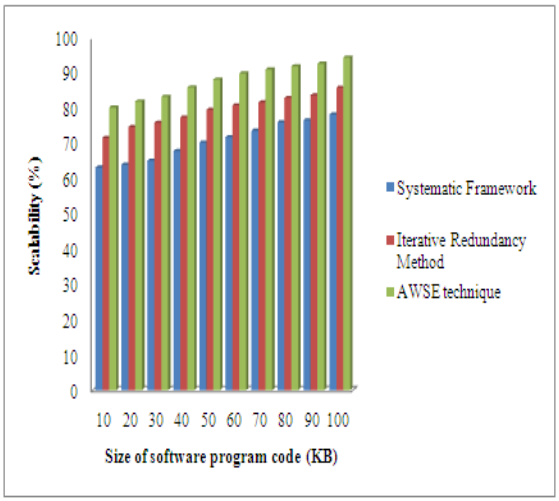

Figure 4 Measurement of scalability

Figure 4 depicts the impact of scalability versus diverse size of software code using three methods. As exposed in figure, the proposed AWSE technique provides better scalability for improving the performance of software quality management when compared to existing systematic framework [1] and Iterative redundancy method [2]. In addition, while increasing the size of open source software program code, the scalability is also increased using all the three methods. But comparatively, the scalability using proposed AWSE technique is higher. This is due to application of

4.2 Measurement of Service provisioning time

In AWSE technique, Service provisioning time (SPT) measures the amount of time required to creating the contemporary version of open source application with the conventional versions to render the user requirements. The service provisioning time is measured in terms of milliseconds (ms) and mathematically formulated as,

$$SPT = P_{size} * time\,(creating\ contemporary\ version\ of\ open\ source\ application) \quad (16)$$

From the equation (16), service provisioning time of software quality management is obtained. While the service provisioning time is lower, the method is said to be more efficient.

Table 2 Tabulation for Service provisioning time

Size of software program code (KB)	Service provisioning time (ms)		
	Systematic Framework	Iterative Redundancy Method	AWSE technique
10	27.8	19.6	11.2
20	30.2	25.3	14.5
30	35.4	29.7	19.3
40	38.9	33.1	22.7
50	41.5	36.4	25.8
60	44.7	42.5	28.3
70	49.3	45.8	30.4
80	55.8	50.2	35.1
90	59.2	56.7	39.9
100	65.5	62.9	45.3

Table 2 illustrates the comparative result analysis of service provisioning time based on diverse size of software program code using three methods. While 30 KB open source software code is taken for software quality management, proposed AWSE technique acquires the 19.3 ms service provisioning time whereas the systematic framework [1] and Iterative redundancy method [2] acquires 35.4ms and 29.7ms respectively.

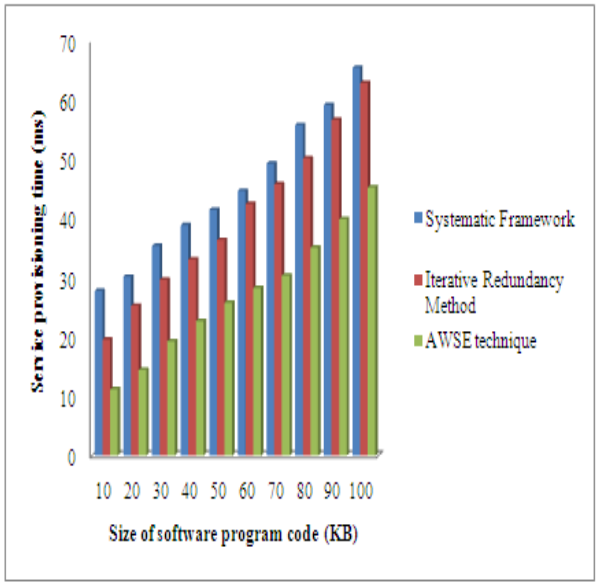

Figure 5 Measurement of Service provisioning time

Figure 5 portrays the impact of service provisioning time versus different size of software code using three methods. As demonstrated in figure, the proposed AWSE technique provides better service provisioning time for improving the reliability of software quality in software quality management when compared to existing systematic framework [1] and Iterative redundancy method [2]. As well, while increasing the size of open source software program code, the service provisioning time is also increased using all the three methods. But comparatively, the service provisioning time using proposed AWSE technique is lower. This is because of application of autocorrelation function in which association between the contemporary version and conventional versions of same open source software is efficiently determined for updating the conventional versions. Furthermore, autocorrelation for service provisioning time entropy is estimated to find out the amount of time required for constructing the contemporary version of open source application with the conventional versions. This in turn helps for reducing the service provisioning time of software quality management. Thus, the proposed AWSE technique reduces the service provisioning time by 41% as compared to systematic framework [1] and 11% as compared to Iterative redundancy method [2] respectively.

4.3 Measurement of Software reliability

In AWSE technique, Software reliability measures possibility of failure-free software operation using equation (8) in software program code. The Software reliability is measured in terms of percentage (%). While the software reliability is higher, the method is said to be more efficient.

Table 3 Tabulation for Software reliability

Size of software program code (KB)	Software reliability (%)		
	Systematic Framework	Iterative Redundancy Method	AWSE technique
10	64.89	70.12	85.36
20	65.22	70.95	86.25
30	65.91	73.62	86.91
40	68.21	74.48	88.15
50	68.95	75.86	88.89
60	70.88	77.65	89.65
70	71.25	78.21	90.50
80	72.93	78.90	91.42
90	75.16	79.46	92.80
100	76.38	81.20	93.95

The software reliability result is obtained with respect to different size of software program code using three methods is presented in Table 3. While 80 KB open source software code is taken for improving software quality, proposed AWSE technique obtains the 91.42 % software reliability whereas the systematic framework [1] and Iterative redundancy method [2] obtains 72.93% and 78.90% respectively.

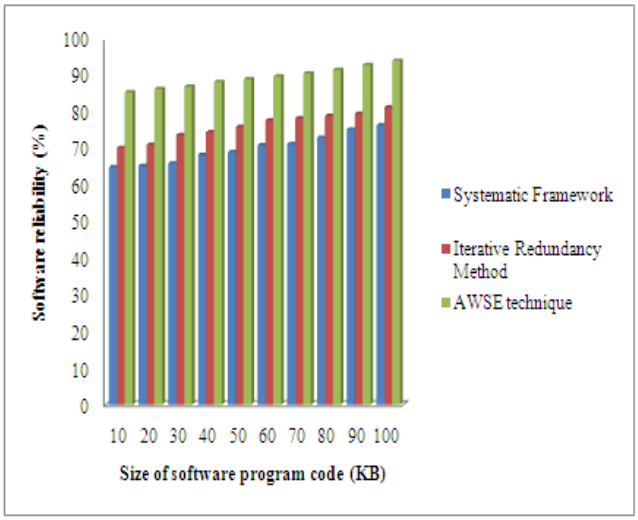

Figure 6 Measurement of Software reliability

Figure 6 describes the impact of software reliability versus diverse size of software code using three methods. As revealed in figure, the proposed AWSE technique provides better software reliability when compared to existing. Further, while increasing the size of open source software program code, the software reliability is also increased using all the three methods. But comparatively, the software reliability using proposed AWSE technique is higher. This is because of application of weighted sum entropy model in AWSE technique. The weighted sum entropy model efficiently finds the faults of given open source software and mean time between the failures with objective of enhancing the quality of software. This in turns assists for improving the software reliability. Hence, the proposed AWSE technique increases the reliability of open source software by 28% as compared to systematic framework [1] and 18% as compared to Iterative redundancy method [2] respectively.

5. RELATED WORKS

In [11], the impact of different software metrics designed for evaluating software quality was analyzed to improve the reliability of software system. But, Reducing the time and cost of the software project was not considered. Modularity Index Metrics was developed in [12] for Java-Based Open Source Software Projects to discover strengths and potential problems of the project.

In [13], a collection of metrics was presented to determine the impact of metrics in software development environment and investigated the open source tools for automation of metrics generation process. A review of different techniques designed for software quality models to evaluate the software products was presented in [14].

An empirical approach was developed in [15] to learn software metrics impacts on different versions of Java based open source software's. However, human and environmental factors which affect maintainability of open source software's were remained unaddressed.

An analysis of code ownership metrics and their relationship with software quality was presented in [16] to improve the reliability of open source software projects. But, service provisioning time was remained unsolved.

A survey of different static and dynamic metrics designed for Open Source Software was analyzed in [17] to assure software code quality, operation, and maintenance. Besides, a survey of diverse open source tools developed for measuring the internal quality of Java software products was examined in [18].

A fuzzy data mining algorithm was intended in [19] for time series data to generate the association rules for evaluating the existing trend and regularity in the evolution of open source software project. Quality Management for achieving higher software quality in the Service Sector was presented in [20].

6. CONCLUSION

An effective component model called Autocorrelation Weighted Sum Entropy (AWSE) technique is developed with objective of attaining the higher scalability and reducing service provisioning time while improving the reliability of software quality. The main objective of AWSE technique is to improve the software reliability with reduced cost of software quality testing. This objective of AWSE technique is attained with application of autocorrelation function and weighted sum entropy model. At first, AWSE technique employed autocorrelation function to find correlation among the consecutive version of same open source software which in turn helps for enhancing the scalability and reducing service provisioning time of software quality management. After that, AWSE technique used weighted sum entropy model that measures the Mean Time between Failure (MTF) to improve the software reliability in a specified environment for a given amount of time. Further, AWSE technique integrates the two objectives (i.e., cost minimization and testing time minimization) into a single objective through multiplying each objective with a weight. This in turn helps for reducing the total cost of software quality testing. The efficiency of AWSE technique is test with the metrics such as scalability, service provisioning time and software reliability. With the experiments conducted for AWSE technique, it is observed that the scalability for improving software system quality provided more accurate results as compared to state-of-the-art works. The experimental results demonstrate that AWSE technique is provides better performance with an improvement of software reliability and also reduces the software reliability when compared to the state-of-the-art works.

REFERENCES

[1] Ivan Janicijevic, Maja Krsmanovic, Nedeljko Zivkovic, Sasa Lazarevic, "Software quality improvement: a model based on managing factors impacting software quality", Software Quality Journal, Springer, Volume 24, Issue 2, pp 247–270, June 2016

[2] Yuriy Brun, Jae young Bang, George Edwards, and Nenad Medvidovic, "Self-Adapting Reliability in Distributed Software Systems", IEEE Transactions on Software Engineering, Volume: 41, Issue 8, Pages 764 – 780, 2015

[3] Julio Escribano-Barreno, Javier García-Muñoz and Marisol García-Valls, "Integrated Metrics Handling in Open Source Software Quality Management Platforms", Advances in Intelligent Systems and Computingm Springer, Pages 509-518, May 2016

[4] Jakob Axelsson, Mats Skoglund, "Quality assurance in software ecosystems: A systematic literature mapping and research agenda", The Journal of Systems & Software, Elsevier, Volume 114, Pages 69–81, April 2016

[5] Vinay Tiwari, R.K. Pandey, "Open Source Software and Reliability Metrics", International Journal of Advanced Research in Computer and Communication Engineering, Volume 1, Issue 10, Pages 808-815, December 2012

[6] Adewole Adewumi, Sanjay Misra1, Nicholas Omoregbe, Broderick Crawford and Ricardo Soto, "A systematic literature review of open source software quality assessment models", Springer, Volume 5, Issue 1, Pages 1-13, 2016

[7] Foutse Khomh, Bram Adams, Tejinder Dhaliwal, Ying Zou, "understanding the impact of rapid releases on software quality The case of firefox", Empir Software Engineering, Springer, Volume 20, Issue 2, Pages 336–373, May 2014

[8] Salah Bouktif, Houari Sahraoui, Faheem Ahmed, "Predicting Stability of Open-Source Software Systems Using Combination of Bayesian Classifiers", ACM Transactions on Management Information Systems, Volume 5, Issue 1, Pages 1-25, 2014

[9] Nicholas Drouin, Mourad Badri, and Fadel Toure, "Metrics and Software Quality Evolution: A Case Study onOpen Source Software", International Journal of Computer Theory and Engineering, Volume 5, Issues 3, Pages 523-527, June 2013

[10] Shelbi Joseph, Akhil P, Seetha Parameswaran, "Reliability Estimation of Open Source Software based Computational Systems", International Journal of Innovative Research in Computer and Communication Engineering, Volume 5, Issue 2, Pages 1310-1317, February 2017

[11] Mrinal Singh Rawat, Arpita Mittal, Sanjay Kumar Dubey, "Survey on Impact of Software Metrics on Software Quality", International Journal of Advanced Computer Science and Applications, Volume 3, Issue 1,Pages 137-141, 2012

[12] Andi Wahju Rahardjo Emanuel, Retantyo Wardoyo, Jazi Eko Istiyanto, "Modularity Index Metrics for Java-Based Open Source Software Projects", International Journal of Advanced Computer Science and Applications, Volume 2, Issue 11, Pages 52-58, 2011

[13] Shefali Singla, Dheerendra Singh, "Classification of Software Metrics and Open Source Tools for Software Development Phase", International Journal of Computer Science & Communication, Volume 5, Pages 103-109, 2014

[14] José P. Miguel, David Mauricio and Glen Rodríguez, "Review of Software Quality Models for the Evaluation of Software Products", International Journal of Software Engineering & Applications (IJSEA), Volume 5, Issues 6, Pages 31-53, November 2014

[15] Mukti Chauhan, Monika Sharma, Predicting Maintainability of Open Source Softwares: An Empirical Approach", International Journal of Engineering Research & Technology (IJERT), Volume 2, Issue 6, Pages 3333-3336, June – 2013

[16] Matthieu Foucault, Cedric Teyton, David Lo, Xavier Blanc, Jean-Reemy Falleri, "On the usefulness of ownership metrics in open-source software projects", Information and Software Technology, Volume 64, Pages 102–112, 2015

[17] Ankush Vesra, Rahul, "Study of Various Static and Dynamic Metrics for Open Source Software", International Journal of Computer Applications (0975 – 8887), Volume 122, Issue 10, Pages 17-21, July 2015

[18] P. Tomas, M.J. Escalona, M. Mejias, "Open source tools for measuring the Internal Quality of Java software products. A survey", Computer Standards & Interfaces, Elsevier, Volume 36, Pages 244–255, 2013

[19] Munish Saini, Sandeep Mehmi, and Kuljit Kaur Chahal, "Understanding Open Source Software Evolution Using Fuzzy Data Mining Algorithm for Time Series Data", Hindawi Publishing Corporation, Advances in Fuzzy Systems, Volume 2016, Article ID 1479692, Pages 1-13, 2016

[20] Radoslav Jankal, "Software Support of Quality Management in the Service Sector", Procedia - Social and Behavioral Sciences, Elsevier, Volume 149, Pages 443 – 448, 2014