ISSN 2278-3091
Volume 9, No.3, May - June 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse184932020.pdf
https://doi.org/10.30534/ijatcse/2020/184932020

Component-Based Software System using Computational Intelligence Technique for Reliability Prediction

Shivani Yadav[1], Bal Kishan[2]
Department of Computer Science and Applications, Maharshi Dayanand University, Rohtak-124001, India
[1]shivaniyadav17@gmail.com
[2]balkishan248@gmail.com

ABSTRACT

Software developers and companies are working to create reliable software for their users. This has resulted in the need for enhancing predictive software quality models for risk-less software development based on the type of software. Development and testing costs can be reduced by reusing existing codes and components. Re-use of components results in increased productivity, quality, and maintainability of the software as iterations are made for incremental changes, new developments altogether, and reducing development cost, time, effort. Component-based software systems along with soft computing techniques should be used for having more reliable software. With the increase in demand for software quality prediction, several techniques have been used to predict software quality. The focus of this paper is on the software product's quality; metrics and we thoroughly review the literature of existing computational intelligence/soft computing techniques and analyze the findings according to the techniques. We have discussed models for software quality prediction for component-based software and computational intelligence techniques which are widely prevalent for software quality prediction and compare different techniques. In our study, we have done a comparative analysis of soft computing techniques. This study will help the other researchers to study and understand which technique would be more helpful in making the software reliability prediction model for component-based software by combining different computational techniques.

Key words: Reliability, models, soft computing, CBS

1. INTRODUCTION

IEEE Standard Glossary of Software Engineering Terminology [1, 2], defines the software product's quality in two ways: 1) the extent to which development of software, system or any component meets the desired condition and 2) if the developed software, system, or component satisfies the user prospect.

Software failures can originate during different software development lifecycle phases as a result of the work done by designers, programmers, and analysts. Software testing ensures that the software is reliable and free of any such residual errors. Modeling of testing scenarios to cover all possible test cases is an integral part of this exercise and requires tremendous work. Once developed, these models successfully are used for error prediction and estimation, serving as the basis for software reliability and the need for further testing to prevent all current and future failure scenarios.

Software quality has been measured using multiple models that use various characteristics and their relationships specifying quality requirements. These characteristics can be thought of as fundamental factors, each of which can have sub-factors). Metric based evaluation can be done for these sub-factors of software quality. [3]

There is a lot of common vocabulary for defining quality concepts for software product development and goods manufacturing. Due to the complex and intellectual nature of large software systems, the development of software based on components is gaining traction. The measurement of quality and reliability aspects of component-based software products requires its own set of tools, which can help in improving the productivity of developers as well.

One of the major challenges for component-based system developers is selecting the right component from a multitude of components available in the marketplace. It is difficult to define a single uniform criterion for optimal component selection. The process of component selection involves information gathering, modeling, selection, simulation of the system, etc. which makes it a long and tedious process. Any changes to the services of a component require the same set of processes to be re-run for the re-examination of the system. To overcome the time challenge, developers are prone to accepting the first working solution instead of running the same configuration over a range of components to find the optimal component [4-6]. These choices can often affect system performance and product design. Automation of the detailed selection process provides a good opportunity for automation to aid in the examination of components in a short period. As a result, there are multiple approaches for component selection using computational intelligence techniques for increasing the overall reliability of the system.

Soft computing techniques try to solve real-world problems, characterized by uncertain, imprecise, and difficult to categorize. It could be seen as analogous to working with natural elements like plants and animals which are adaptive and flexible in behavior [7].

Here, soft computing approaches have been introduced, which examines functional and non-functional, requirements for determining the optimal components required for a system. The prediction models could be based on some computational intelligence approaches likewise Regression Models, Artificial Neural networks, and Fuzzy systems [8]. While traditional quality metrics consisted of functions that ingest software data and provided a single value, denoting the degree of an attribute measuring software quality, soft computing methods estimates provide estimated development cost and effort using computational intelligence/ soft computing optimization algorithms.

1.1 Software Quality Metrics

Let us have a look at some of the major quality metrics:

Maintainability: Three quality metrics are generally used to measure the maintainability of software: instability, abstractness, and efferent coupling. The application which is having an abstractness(A) value equal to 0 and instability(I) value equal to 1, known as non-dependable application and application having efferent coupling(C_e) equal to 0 and instability(I) equal to 0 is categorized as a reliable application. Abstractness is computed with the help of the subsequent expression,

$$Abstractness = N_a/N_c \text{ (9), where}$$

N_a = abstract classes
N_c = no. of total classes

Furthermore, instability computed as,

$$(C_e) / (C_{e+} C_a) \text{ (10), where}$$

C_e = Efferent Coupling
C_a = Total Coupling

Another method to calculate the maintainability of software is to find the number of dependable applications that make up the software. Higher the number of dependable application components, the lower will be its maintainability.

Reliability

A key parameter for estimating the quality of the software is its reliability. The ability of software to be accessible and provide failure-free operations is called reliability. It can be determined at implementation time by working out the failure rate and execution time. Reliability can be measured using the following expression,

$$Reliability = \frac{Failure\ rate}{Execution\ time} \quad (11)$$

Reusability: Reusability is the degree to which components or features of the software can be reused to create new software applications. The aim is to avoid repetition of work and save development efforts, costs, and time. Since the components continue to be reused and tested over time, the reliability of the systems also tends to be higher. Reusability of software components can be assessed using factors like,

- Understandability – ease of understanding the source code and functionality of the component
- Portability – ease of adoption across different environments

1.2 Quality Model

There are two major categories of quality models:
1. Basic Models – these models focus on complete and comprehensive product evaluation
2. Tailored Quality Models – which focus on the evaluation of components

The Basic Models can be used for any kind of software products and are hierarchical in structure. The six most important are: Mc Call[1977], Boehm [1978], FURPS [1992], Dromey [1995], ISO 9126-1 [2001]; its variants for internal (ISO / IEC 9126-3 [2003]), external (ISO / IEC 9126-2 [2003]), and metrics used in quality: ISO / IEC 9126-4 [2004]. The ISO -9126 model as a result of inputs from previous models, namely Boehm and McCall models. A new adapted model: ISO 25010 or ISO/IEC CD 25010 was established in 2007 [3] called as Software product Quality Requirements and Evaluation or SQuare.

Tailored Quality Models started coming up in 2001 with the Bertoia model. Further models were proposed by Georgiadou[2003], Alvaro[2005], and Rawashdesh. They differ from basic models since their applicability is confined to specific application domains, where the relative importance of features may vary according to that particular domain. The need for such models is driven by specialized organizations and their need for evaluation of individual components. Most of these models are adapted from Basic Models, with a slight modification to fulfill the goals of different domains.

ISO 9126 MODEL: This model is an enhancement of Boehm and McCall models as ISO 9126 model uses the basics concepts of both the models mentioned above. It focuses on two aspects
1) The characteristics of internal and external quality, and
2) The quality of user characteristics mentioned.

Internal quality characteristics can be evaluated without execution, like source code, while external attributes require execution to be evaluated. External quality attributes can only be assessed during system operation or maintenance phase.

The quality in use attributes is concerned with the efficiency, effectiveness, and security of the software product and the resulting user satisfaction. The ISO-9126 model brought about standardization in software quality terminology and used as the ground model for building tailored quality models. Its main aspect was to institutionalize the phrasing of the software

quality [3]. ISO 9126 quality model for internal and external quality incorporate reliability, functionality, efficiency, usability, portability, and maintainability. Furthermore, quality in use provides effectiveness, productivity, safety, and satisfaction.

ISO 25010 Model: ISO 25010 emerged in 2007, as an update of the existing ISO 9126 model. According to this model, software product quality can be divided into 8 key features and each feature has some specified characteristics. The aim of this model is quality driven software development.

One major change in this model is the removal of portability as a key feature. Instead, security and compatibility have been used to encompass some characteristics previously considered part of portability, and also some other characteristics that weren't considered earlier. Portability has been clubbed under Transferability as a characteristic.

ISO 25010 shares similarities with the ISO/IEC 9126 model along the lines of internal, external, and quality in use attributes. Parameters used for software quality are reliability, performance, security, maintainability, transferability, compatibility, operability, and functional suitability [12].

1.3 Soft Computing Techniques

Many new algorithms which emulate natural process optimization have surfaced over time, including genetic algorithm by Holland in 1975, simulated annealing by Kirkpatrick, Gelatt Jr., and Vecchi in 1983, evolutionary algorithms by Schwefel in 1995, ant colony optimization by Dorigo and Maria in 1997 and particle swarm optimization by Parsopoulos and Vrahatis in 2002. All these algorithms provide incremental improvement by incorporating current inputs and finding a more optimal solution in the search space. Since these algorithms do not take derivates of the cost function, they can be applied to discrete variables and non-continuous cost functions as well. In figure 1, we showcase some of the important prediction techniques of the software quality deployed on soft computing techniques/ computational intelligence technique and their categorization

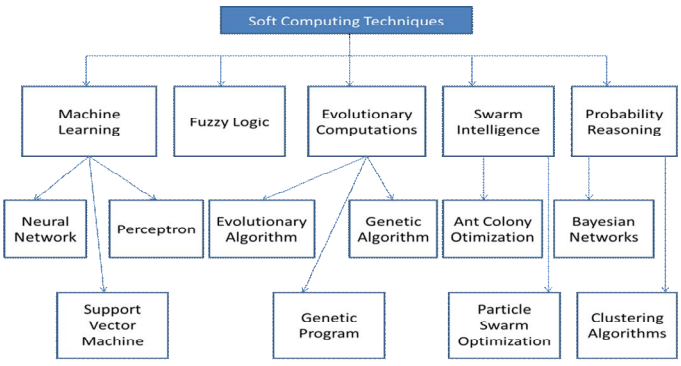

Figure 1: Different Soft Computing Approaches

Neural Networks: Neural network is defined as a circuit containing many simple processing elements based on neutrally simple processing elements. The building blocks operate on local information in an asynchronous manner, thereby removing the need for an overall system clock.

Fuzzy logic: The concept was developed by Zadeh at UC Berkeley for representing data that is imprecise as a result of its natural behavior. It allows transitional, also called fuzzy, values between crisp or non-fuzzy values like yes/no, high/low, etc. In the area of designing non-linear control systems, fuzzy logic plays an important role.

Evolutionary computing: Evolutionary computing can be viewed as the optimization methods and stochastic search and algorithms computed from the natural theory of progression or Theory of Darwinian [13, 14]. These algorithms have found applications across numerous research areas, such as bioinformatics.

Bayesian network: These are models that show the graphical representation for probabilistic computations based on the Bayesian theory of probability. In a Bayesian network, the nodes show continuous or discrete variables along with curves producing connections among variables. It can find applications across fields like document classification, image processing, and medicine.

Chaos theory: Chaos theory focuses on deterministic systems that are highly dependent on the initial conditions. It states that, while the events may appear chaotic on the horizon, but are quite structured and one can find underlying patterns and triggers that result in a change in the state of non-linear systems. A small change, no matter how minuscule, in the state of a deterministic non-linear system can result in trajectories that are exponentially different over some time. However, the system needs to be at least three dimensional and non-linear [15].

Genetic Algorithm: Genetic algorithms maintain a population set consisting of individuals, represented as a set of chromosomes. Fitness scores are assigned to each individual based on their degree of satisfaction with the desired criterion. A new generation is created by a three-step process:
Selection – individuals with better fitness scores are selected and allowed to pass on their chromosomes to future generations.
Crossover – chromosomes from selected individuals are combined to create completely new individuals
Mutation – a random variation is introduced into chromosomes of a new generation

This process is repeated until the required level of accuracy or optimization is achieved. The model is derived from the theory of evolution, where individuals might expire but the remaining population achieves fitness over some time. Genetic algorithms are used to solve optimization issues for complex, non-linear problems using machine learning models [16].

2. LITERATURE REVIEW

This section discusses different techniques/methods of predicting different software quality models based on component-based software combining with computational techniques for the enhancement of the quality of software.

Kavita Sheoran and Om Parkash Sangwan used existing models for software quality prediction. The results from the Software quality model (SQM) were compared with ISO 25010, Component-based quality model (CBQM), ISO 9126, Bertoia, and Alvaro model. The study was conducted using secondary sources of data. Several characteristics were considered, like reliability, usability, maintainability, and portability, along with sub-characteristics like understandability, performance, compatibility, and accuracy. It was found that the Alvaro model was able to provide better results, especially in terms of accuracy, testability, and understandability [17].

Mamta Punia and Amandeep Kaur put forward a method to predict software maintainability levels on a five-level scale, ranging from very well to very poor, using soft computing techniques and MATLAB's fuzzy logic toolbox. The toolbox helped create rules and generate training and test data sets, which were then fed into a multilayer feed-forward neural networks. The method was evaluated using Mean Absolute Relative Error (MARE) and Mean Relative Error (MRE). The experimental conclusion showed reasonable levels of accuracy and usefulness of an artificial neural network (ANN) in predicting software maintainability [18].

Deepak Gupta *et al.* discussed a study using multiple estimation techniques of software quality, including Fuzzy System, Regression Tree, Multiple Linear Regression, Rule-Based System, Case Base Rule, and Artificial Neural Network, and their respective performance. The aim was to construct an accurate software quality prediction model. The best results were obtained from the fuzzy and rule-based system but no single technique could alone fulfill all requirements and emphasized the need for hybrid techniques [19].

Sheikh Fahad Ahmad conducted a comparative analysis of software quality models and various metrics associated with those models for predicting software reliability. Characteristics like size, performance, complexity, quality, etc. were considered for evaluation using three proposed models: McCALL, BOEHM, and ISO9126 [20].

Tibor Bakota *et al*. sought to build a probabilistic approach that could use expert knowledge to deal with imprecision while computing complex quality characteristics. It used the freedom offered by ISO9126 standard to propose a new approach while focusing on maintainability. An acyclic graph with nodes corresponding to inward-looking (source code) and outward-looking (execution performance) quality properties was constructed to determine quality characteristics. The measures of these characteristics were expressed as a goodness function on an interval scale, where 0 and 1 are the worst and best cases. The results showed changes in the quality model with the occurrence of maintenance activities in a positive correlation. Development activities could be revealed by changes in values [21].

Mbusi Sibisi *et al.* created a framework for quality requirement specification and defined the characteristics in ISO/IEC 9126-1 (2001). The research focused on creating a framework for adapting software quality models that could work on an intermediate or end software product and meet different customer and organizational needs. While a general quality profile questionnaire is used to select reliable metrics and rating levels, it requires an objective approach to select appropriate characteristics and sub-characteristics at the product level. Results were validated by focusing on seven factors listed in ISO 9126-2: Reliability, Repeatability, Reproducibility, Availability, Inductiveness, Correctness, and Meaningfulness. It was found that the validation process was successful at the characteristic level, whereas, sub-characteristics level validity required further improvements [22].

José P. Miguel *et al*. took a user-centric approach for proposing models to identify quality issues leading to some new measures such as reusability, configurability, availability, lower cost, and better quality, were considered for evaluating the components. Some of the models, with a range in a small domain, have been adapted from ISO 9126. Basic models can also be adapted to build custom quality models as per requirement. Open-source models highlight community-driven requirements [3].

Ashwin B. Tomar et al. evaluated some quality models based on their methodology and techniques used. The models were evaluated using case studies and experiments, apart from expert opinions and surveys. The authors went through seventy research papers on software quality, before categorizing them into eight research areas. The paper recommended the need for further research on software quality models [23].

Hsu et al. proposed an adaptive model for path reliability estimation testing for component-based software systems. The model could use three methods: brand, sequence, and loop, for path reliability estimation. According to the author, the resulting path reliability model could estimate the application reliability [24].

Diwaker et al. estimated the usability of reusable components and system integration using the interaction between components in the purview of Component-Based Systems. Reusability ensures a better estimation of efficiency and reliability over time. According to the author, Ant Colony Optimization was used to identify reusable components and interaction of components [25].

Mohamed Abdullahi Ali *et al.* conducted a literature review and proposed a component quality model to determine the characteristics of a good component. The literature review was

carried out using protocol-based automated searches. According to the author, the proposed quality model had design phase-specific metrics and was highly relevant for addressing design issues [26].

Olusola used the Genetic-Fuzzy system to evaluate the reusability of 69 software components using five quality factors. Software components were selected from 3rd party software vendors and data extracted from those components was used to compute metric values of the five quality factors. The results of the Genetic-Fuzzy System (GFS) approach were used for the comparison with the Adaptive Neuro-Fuzzy Inference System (ANFIS) technique with the help of an average root-mean-square error. According to the author, the GFS approach provided better accuracy than ANFIS. Another finding was the better suitability of Java components for reuse than the other components used in the experiment [27].

Osheen Bhardwaj *et al.* analyzed different techniques presented in research papers for estimated quality in component-based development. According to the author, important perspectives which they identified are significant for reducing time, effort, and cost for development through reuse. These are Improvements in the component-based framework by reducing its complexity. The authors have also suggested a monkey testing approach for security improvement and quality maintenance [28].

Hu et al. used modified adaptive testing for developing a model for estimating the reliability of component-based systems. According to the authors, they concluded the importance of failures observed from a user's side through extended metrics placed on Nelson's reliability model. An adaptive model was built to select test cases, within a limited budget, based on test history information. This enhanced the testing process by improved test case selection [29].

Lance Fiondella *et al.* introduced a model which is based on the concept of Correlated Component Failures (COCOF). Providing due consideration to application architecture, correlation and, component reliabilities, and efficient software reliability assessment approach is proposed. According to the author, the algorithm used in the approach is transforming a Multivariate Bernoulli distribution into a joint distribution of component outcomes [30].

Studying all the previous work done by various researchers helps us to assess that there is a critical need to work in the field of making reliable software in fixed time and cost asked customers/organizations. The quality of software can be increased through many ways like testing each component at every phase of the software development life cycle (early removal of bugs), using various optimization techniques [31-38]. Nowadays, a reliable, more efficient, and fault-free software product is demanded in almost every field like routing protocols in the area of communication, big data analysis in the area of security of web, social networking sites, facial recognition and for working in a smart environment, etc [39-49]. But, my work focuses on making an early fault predicting reliability model based on component-based software using computational intelligence techniques. Computational intelligence techniques provide us transcendent results with a large amount of data.

Table 1 below describes the various quality models, parameters supported by them, approaches used by the model, functional/non-functional behavior, its advantages, and disadvantages.

Table 1: Various quality models, parameters supported

Model	Parameters Supported	Approach	Functional/ Non Functional	Advantages	Disadvantages
FURPS	Functionality, Reliability, Usability, Supportability, Performance	Component, Object-oriented, Hierarchical	Functional and Non-Functional	Clearly defines functional and non-functional requirements	• Does not consider account portability and maintainability • Only takes into account user requirements and disregards developer considerations • Doesn't take into account domain-specific attributes
Dromey's Quality Model	Functionality, Reliability, Usability, Reusability, Portability, Efficiency	Component	Functional and Non-functional	• Broad enough to work for different systems • Added reusability and process maturity	Software quality measurement criteria missing
Bertoa's Quality Model	Functionality, Reliability, Usability, Efficiency, Maintainability	Component		Allows for effective evaluation of COTS products	• Does not address portability and reusability • Incomplete due to failure of carrying out an experiential assessment
McCalls Model	Correctness, Reliability, Efficiency, Integrity, Usability, Maintainability, Flexibility, Testability, Portability, Reusability, Interoperability	Object-oriented, Hierarchical	Non-functional	Quality characteristics and metrics relationship	• The functionality of software products not considered directly • Difficult to use this framework to set precise and specific quality requirements as it is based on Yes/No responses
Ghezzi Model	Integrity, Flexibility, Accuracy, Portability, Maintainability, Reliability, Usability, Reusability	Component	Functional and Non-functional	Helps developers achieve external and internal qualities	Dependent on internal software quality and developer
IEEE Model	Efficiency, Functionality, Usability, Reliability, Portability, Maintainability	Component	Functional and Non-functional	Provides a standard for software maintenance and includes other standards such as software quality assurance	Describes only qualitative factors of various measurement techniques
ISO 9126-1Model	Accuracy, Efficiency, Functionality, Interoperability, Maintainability, Portability, Security, Usability	Object-oriented, Hierarchical	Functional and Non-functional	• Applicable to every kind of software • Identifies internal and external quality characteristics	Generality
Boehm Quality Model	Flexibility, Reliability, Portability, Efficiency, Testability, Understandability, Usability, As-is Utility	Object-oriented, Hierarchical	Functional and Non-functional	• Hierarchical representation of software product characteristics to get a contribution in total quality.	• The suggestion about measuring the quality characteristics missing • Architectural integrity is not considered

Over the years, a multitude of reliability models has been conceptualized, analyzed, and evaluated. Software reliability growth models have been greatly improved with the advent of soft computing techniques like Fuzzy Logic, Genetic Programming (GP), Neural Network (NN), Ant Colony Optimization (ACO), Genetic Algorithms (GA), and Artificial Bee Colony (ABC), etc. have been summarized in table 2 below:

Table 2: Different Soft Computing Techniques

Sr. No.	Title of the Paper	Author's Name	Methodology Used	Objective	Summary/Findings
1.	"A Study of the Connectionist Models for Software Reliability Prediction" [50]	'S. L. Ho, M. XIE and T. N. GOH'	Neural network	To review the usefulness of a modified Elman recurrent neural network in predicting software failures	The Elman model yields slightly better results than the Jordan model and has a significant advantage over the feed-forward model
2.	"Employing four ANNs Paradigms for Software Reliability Prediction: an Analytical Study" [51]	'Sultan H. Aljahdali and Khalid A. Buragga'	Neural network	To analyze the performance of four different connectionist paradigms reliability prediction modeling	Due to the capture of the changing nature of the used data set, improved function prediction using Elman recurrent NNs and achieves better prediction capability
3	"Software Reliability Prediction using Neural Network with Encoded Input" [52]	'ManjubalaBisi and Neeraj Kumar Goyal'	Neural network	To draw up guidelines for encoding parameter identification which provides consistency in results across different datasets	Proposed a feed-forward neural network which uses exponential and logarithmic functions for encoding scheme and results after comparison delivered good prediction capability
4	"Estimation for Faults Prediction from Component-Based Software Design using Feed Forward Neural Networks" [53]	'Sandeep Kumar Jain and Manu Pratap Singh'	Neural network	To predict component reliability using various neural network architectures	Estimated fault prediction behavior for a complete software product and a subset of components over a cumulative execution time
5	"Application of Fuzzy Time Series in Prediction of Time Between Failures & Faults in Software Reliability Assessment" [54]	'S. Chatterjee, S. Nigam, J.B. Singh, and L.N. Upadhyaya'	Fuzzy Logic	To linguistically express software failure for reliability model validation	Proposed models are flexible and computationally simple, with reduced execution time. They do not require any de-fuzzified techniques
6	"Development of Software Reliability Growth Models for Industrial Applications Using Fuzzy Logic" [55]	'Sultan Aljahdali'	Fuzzy Logic	To explore the usability of fuzzy logic for development of SRGM for fault estimation during testing	Developed high-performance modeling capable models

7	"Software Reliability Modeling Using Soft Computing Technique" [56]	'KhatatnehKhalaf and Thaer Mustafa'	Fuzzy Logic	To develop an accurate model using a custom set of test data through a fuzzy logic technique	Accurate predictions for the target database using developed models
8	"A Genetic Programming Approach for Software Reliability Modeling" [57]	'Eduardo Oliveira Costa, Aurora Trinidad Ramirez Pozo, and Silvia Regina Vergilio'	GP	To introduce a cost-efficient technique named (μ+λ) GP (Genetic Programming) for reliability modeling	Proposed (μ+λ) GP system yielded better results than classical techniques for small datasets
9	"A New Software Reliability Growth Model: Genetic-Programming-Based Approach" [58]	'Zainab Al- Rahamneh, Mohammad Reyalat, Alaa F. Sheta, SuliemanBani-Ahmad, Saleh Al- Qqeili'	GP	To propose a genetic programming model to develop a Software Reliability Growth Models (SRGM) which predicts faults during the testing process	Genetic Programming operators were recalibrated to boost convergence process
10	"SRGM with Imperfect Debugging by Genetic Algorithms" [59]	'R. SatyaPrasad, O. NagaRaju and R. R. L Kantam'	Genetic Algorithm	To estimate the effect on SRGM after incorporation of a change-point problem and imperfect debugging	The proposed model with the exponential distribution finds better software reliability as compared to other existing models
11	"The Research on Reliability Optimization of Software System Based on Niche Genetic Algorithm" [60]	'Qian Yuexia, and GuWeijie'	Genetic Algorithm	To propose a novel Genetic Algorithm for system reliability optimization	The proposed algorithm resolves the reliability of multi-module complex software effectively whilst also improving the speed and quality of resolution
12	"Assessing Software Reliability Using Modified Genetic Algorithm: Inflection SShaped Model" [61]	'R. Satyaprasad, and G. Bharathi'	Genetic Algorithm	To develop a modified genetic algorithm for software reliability assessment by developing Inflection S-shaped model based on the time domain software failure data using	The suggested algorithm works better and quicker than traditional algorithms
13	"Enhancement and comparison of ant colony optimization for software reliability models" [62]	'Latha Shanmugam and Lilly Florence'	Ant Colony	To compare the ant colony optimization method with an enhanced version of the same	Improved Ant Colony Optimization method gave improved estimation accuracy, with reduced time and space complexity
14	"Software Reliability Prediction by Using Ant Colony Optimization Technique" [63]	'Ramakanta Mohanthy, Venkatshwarlu Naik, and Azmath Mubeen'	Ant Colony	To develop a novel approach for optimization of reliability prediction	A combination approach provides better results than traditional Ant

				models using raw data	Colony Optimization method
15	"Software Defect Prediction using Ant Colony Optimization" [64]	'Kiran Kumar B., Dr. JayadevGyani, and Dr. Narsimha G'	Ant Colony	To study the effectiveness of ant colony optimization technique on multiple datasets for defect removal	Ant colony optimization technique gives good results on predictions methods
16	"Estimating Software Reliability Using Ant Colony Optimization Technique with Salesman Problem for Software Process" [65]	'D. Hema Latha and P. Premchand'	Ant Colony	To develop an optimized approach for reliability prediction using Ant Colony Optimization method	ACOT with traveling salesman problem assesses real-time data and provides software reliability solutions
17	"Evolutionary algorithms, simulated annealing, and tabu search: a comparative study" [66]	'Habib Youssef, Sadiq M. Sait and Hakim Adiche'	Simulated Annealing	To comparatively study three popular approximation algorithms for floor planning problem: Genetic Algorithm, Simulated Annealing, and Tabu Search	It was observed that Tabu Search provided better results in terms of solution quality
18	"Simulated Annealing Neural Network for Software Failure Prediction" [67]	'Mohamed Benaddy and Mohamed Wakrim'	Simulated Annealing	To propose a hybrid approach comprising simulated annealing techniques and neural network methods	Proposed adaptive simulated annealing method resulted in faster execution time than the RCGA due to reduced search space
19	"The Determination of Preventive Maintenance using Simulated Annealing Algorithm Based on Weighted Fitness Function" [68]	'Yeny Krista Franty, and BudhiHandoko'	Simulated Annealing	To use a simulated annealing algorithm for drawing up a machine maintenance schedule	The proposed schedule increase reliability while minimizing cost
20	"An adaptive neuro-fuzzy model for estimating the reliability of component-based software systems" [69]	'KirtiTyagi and Arun Sharma'	Neuro-Fuzzy	To come up with an adaptive neuro-fuzzy inference model for estimation of reliability for component-based software system	Reliability evaluation of FIS technique is improved by using ANFIS
21	"Soft Computing Techniques For Enhancing Software Reliability" [70]	'Dhavakumar P, Shankar.S, and VikramPandi M'	Neuro-Fuzzy	to give failure-free access software system in the entire environment for roving software reliability	Provides the best indication of prediction strength of developed fuzzy model for accessing the software reliability
22	"An Analysis of Software Reliability Assessment with Neuro-Fuzzy based Expert Systems" [71]	'BonthuKotaiah, MVS Prasad and R.A. Khan'	Neuro-Fuzzy	To examine the effectiveness of reliability assessment methods using neuro-fuzzy based system	Non-parametric models are more preferabler to parametric models as they give accurate reliability even when historical data is missing

23	"The Use of Cuckoo Search in Estimating the Parameters of Software Reliability Growth Models" [72]	'NajlaAkram AL-Saati and MarwaAbd-AlKareem'	Cuckoo Search	To find better parameters for reliability growth models	Results show that Cuckoo Search outperformed both Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) in finding superior parameters tested with identical datasets
24	"Assessment of distribution system reliability using artificial bee colony algorithm" [26]	'Mukul Dixit, PrasantaKundu and Hitesh R. Jariwala'	Artificial Bee Colony	To propose the methodology for distribution system reliability assessment using Artificial Bee Colony (ABC) algorithm	Artificial Bee Colony algorithm identified optimal values for failure rate and repair time for all distribution segments while also minimizing a penalty cost function
25	"Artificial Bee Colony Algorithm for Reliability Analysis of Engineering Structures" [74]	'Haojin Li, Junjie Li and Fei Kang'	Artificial Bee Colony	To demonstrate with the help of an example that Artificial Bee Colony (ABC) algorithm is more reliable and gives accuracy in reliability analysis of engineering structures	The algorithm can be used to efficiently solve global optimization problems that have continuous variables. It can provide a good measure of reliability index

It can be seen that different models require different soft computing techniques. Observations reveal that researchers prefer the Neural Network approach in reliability models. From an accuracy standpoint, genetic programming provides better results as compared to other computational intelligence techniques. Although, simulated annealing and cuckoo search are not used much. The table data is useful to compare and select relevant soft computing techniques for modeling.

3. PROPOSED WORK

Our work focuses on finding a promising solution to detect faults and defects early in the SDLC for component-based software. Systematic and well-defined milestones are required to achieve the objectives.

We will focus on developing a new hybrid approach for testing CBSE based applications by combining the benefits of the component-based approach and computational based technique. The model will be developed by choosing one of the computational intelligence techniques such as machine learning, fuzzy logic, evolutionary computations, swarm intelligence, probability reasoning. One of these techniques is applied to individual components for making reliable software. Hence, from the given computational intelligence techniques, we will focus on particle swarm optimization based on swarm intelligence technique in a combination of fuzzy logic to retrieve optimal results especially in case of large data sets and complex problems. This proposed technique will help in generating software reliability prediction models for component-based software using computational intelligence methodologies.

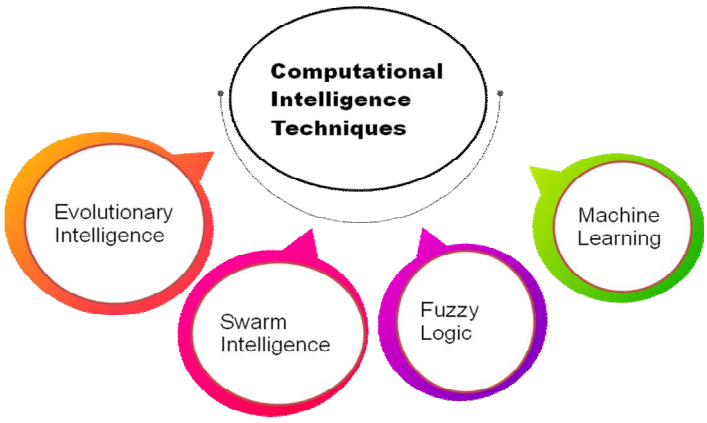

Figure 2: Computing Methodology

4. CONCLUSION

The literature review has shown that there is an abundance of quality prediction models proposed by researchers over the years. However, finding the right model for a particular application is a challenging task as seen by varying degrees of success of different models under different circumstances. There is scope for automating some parameters of measurement offered by quality models, to speed up process and leverage reusability wherever applicable. ISO 25010 has increased the key features to eight, as opposed to six that were present in ISO 9126. Future software quality models will use ISO 25010 as the reference model for product development.

Present shreds of evidence show that soft computing could provide more software quality models to prove software reliability that takes into account the complexity of the task but it needs more work. Models based on computational intelligence techniques such as fuzzy logic, swarm intelligence, machine learning, and evolutionary intelligence have shown promise. When considering quality models for free software, community aspects should be given high priority due to the influence exerted by users in the community, during both, product development and maintenance. In the future, we will choose one of these intelligence techniques for component based software system to build a reliability prediction model, in short we can say a good reliable, fault free quality software.

REFERENCES

[1] IEEE. **Standard for Software Maintenance, Software Engineering Standards** *Subcommittee of the IEEE Computer Society*, 1998.

[2] D. Samadhiya, S. Wang and D. Chen, **Quality Models: Role and Value in Software Engineering,** *Second International Conference on Software Technology and Engineering (ICSTE'10),* pp 320-324, Oct. 2010, doi: 10.1109/ICSTE.2010.5608852.

[3] J. P. Miguel, D. Mauricio and G. Rodriguez, **A Review of Software Quality Models for the Evaluation of Software Products**, *International Journal of Software Engineering & Applications (IJSEA),* vol. 5, no. 6, pp. 31-53, Nov. 2014.

[4] S. Yadav and B. Kishan, **Analysis and Assessment of Existing Software Quality Models to Predict the Reliability of Component-Based Software**, *International journal of emerging trends in engineering research*, vol. 8, no. 6, 2020.[*In Press*]

[5] S. Yadav and B. Kishan, **Reliability of Component-Based Systems – A Review**, *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 8, no. 2, pp. 293-299, 2019. doi.org/10.30534/ijatcse/2019/31822019

[6] S. Yadav and B. Kishan, **Assessment of software quality models to measure the effectiveness of software quality parameters for Component Based Software (CBS)**, *Journal of Applied Science and Computations,* vol. 6, no. 4, pp. 2751-2756, 2019.

[7] K. S. Kaswan, S. Choudhary and K. Sharma, **Software Reliability Modeling using Soft Computing Techniques: Critical Review,** *Journal of Information Technology & Software Engineering*, vol. 5, no. 1, pp. 1-9, Apr. 2015.

[8] K. Sheoran, P. Tomar and R. Mishra, **Software Quality Prediction using Hybrid Classifier based on Improved PSO and ANN**, *Journal of Advanced Research in Dynamical and Control Systems*, vol. 9, pp. 3016-3029, 2017.

[9] G Rasool , N Asif , **Software Architecture Recovery**, *World Academy of Science, Engineering and Technology*, vol. 1, no. 4, pp. 939-944, Jan. 2007.

[10] D. Grosser, P. Valtchev and H. A. Sahraoui, **An analogy-based approach for predicting design stability of Java classes**, *Proc. Ninth International Software Metrics Symposium (METRICS'03)*, pp. 1-10, Oct. 2003, doi: 10.1109/METRIC.2003.1232472.

[11] S. Bouktif, D. Azar, D. Precup, H. Sahraoui and B. K´egl, **Improving Rule Set Based Software Quality Prediction: A Genetic Algorithm-based Approach**, *Journal of Object Technology*, vol. 3, no. 4, pp. 227-241, Apr. 2004.

[12] Ritu and O. P. Sangwan, **Software Quality Estimation Using Soft Computing Techniques**, *International Journal of Innovations & Advancement in Computer Science*, vol. 6, no. 5, pp. 195-205, May 2017.

[13] W. Kuo and V.R. Prasad, **An annotated Overview of System- Reliability Optimization**, *IEEE Transaction on Reliability*, vol. 49, issue 2, pp. 176-187, Jun. 2000.

[14] F. Streichert, **Introduction to Evolutionary Algorithms**, Workshop, University of Tubingen, 2002.

[15] P. D. Kumar, S. Shankar and M. V. Pandi, **Soft Computing Techniques For Enhancing Software Reliability**, *International Journal of Latest Trends in Engineering and Technology*, e-ISSN: 2278-621X, pp. 133-140, Apr. 2018.

[16] M. Punia and A. Kaur, **Software Maintainability Assessment Using Soft Computing Techniques: Review**, *International Journal of Research in Information Technology*, vol. 2, issue 8, pp. 52-56, Aug. 2014.

[17] K. Sheoran and O. P. Sangwan, **An Insight of Software Quality Models Applied in Predicting Software Quality attributes: A Comparative Analysis,** *Pub. Fourth International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions)*, pp. 1-5, Feb. 2015, doi: 10.1109/icrito.2015.7359355

[18] M. Punia and A. Kaur **Software Maintainability Prediction using Soft Computing Techniques**, *International Journal of Innovative Science, Engineering & Technology*, vol. 1, issue 9, Nov. 2014.

[19] D. Gupta, V. K. Goyal and H. Mittal, **Comparative Study of Soft Computing Techniques for Software Quality Model**, *International Journal of Software Engineering Research & Practices*, vol.1, issue 1, pp. 33-37, Jan. 2011.

[20]S. F. Ahmad, **A comparative study of software quality models**, *International Journal of Science, Engineering and Technology Research*, vol. 2, issue 1, pp. 172-176, Jan. 2013.

[21]T. Bakota, P. Hegedus, P. Kortvelyesi, R. Ferenc and T. Gyimothy, **A Probabilistic Software Quality Model**, *Proc. Twenty seventh IEEE International Conference on Software Maintenance (ICSM'11)*, pp. 243-252, 2011, doi: 10.1109/ICSM.2011.6080791.

[22]M. Sibisi and C. C. V. Waveren, **A Process Framework for Customizing Software Quality Models**, *Pub. AFRICON*, IEEE, pp. 1-7, 2007, doi: 10.1109/AFRCON.2007.4401495.

[23]A. B.Tomar and V. M. Thakare, **A Systematic Study of Software Quality Models,** *International Journal of Software Engineering & Applications*, vol.2, no.4, pp. 61-70, Oct.2011.

[24]C. Hsu, and C. Huang, "**An adaptive reliability analysis using path testing for complex component based software systems",** *IEEE Trans. Reliab.*, vol. 60, no. 1, pp. 158-170, 2011.

[25]C. Diwaker and P. Tomar, **Assessment of Ant Colony using Component based Software Engineering Metrics**, *Indian Journal of Science and Technology*, vol. 9, no. 44, pp. 1-5, 2016, doi:10.17485/ijst/2016/v9i44/105159

[26]M. A. Ali and Ng Keng Yap, **Software Component Quality Model**, *International Journal of Engineering and Advanced Technology (IJEAT)*, vol. 9, issue 1, pp. 1758-1762, October 2019.

[27]O. Ajayi Olusola, **Evaluating software components reusability using genetic-fuzzy soft computing approach**, *Australian Journal of Science and Technology*, vol. 3, issue 2, 2019.

[28]O. Bhardwaj and S. Kumar Jha, **Quality assurance through soft computing techniques in component based software**, *International Conference On Smart Technologies For Smart Nation (SmartTechCon)*, pp. 277-282, 2017. doi:10.1109/smarttechcon.2017.8358382

[29]Hai Hu, Chang-Hai Jiang, Kai-Yuan Cai, W. Eric Wong, and Aditya P. Mathur, **Enhancing software reliability estimates using modified adaptive *testin,*** *Information and Software Technology Journal Elsevier, vol. 55, issue 2, pp. 288–300, 2013. doi.org/10.1016/j.infsof.2012.08.012*

[30]L. Fiondella, S. Rajasekaran, and S. Gokhale, **Efficient software reliability analysis with correlated component failures**, *IEEE Transactions on Reliability*, vol. 62, issue 1, pp. 244-255, 2013.

[31] O. Dahiya and K. Solanki, S. Dalal, A. Dhankhar, **Regression Testing: Analysis of its Techniques for Test Effectiveness,** *International Journal of advanced trends in computer science and engineering*, vol. 9, No. 1, pp. 737-744, 2020.

[32] O. Dahiya and K. Solanki, **Comprehensive cognizance of Regression Test Case Prioritization Techniques,** *International journal of emerging trends in engineering research*, vol. 7 No. 11, pp. 638-646, 2019.

[33] O. Dahiya and K. Solanki, S. Dalal, A. Dhankhar, **An Exploratory Retrospective Assessment on the Usage of Bio-Inspired Computing Algorithms for Optimization,** *International journal of emerging trends in engineering research*, vol. 8 No. 2, pp. 414-434, 2020.

[34] O. Dahiya and K. Solanki, and A. Dhankhar, **Risk-Based Testing: Identifying, Assessing, Mitigating & Managing Risks Efficiently In Software Testing,** *International Journal of advanced research in engineering and technology*, vol. 11, Issue 3, pp. 192-203, 2020.

[35] K. Solanki, and S. Kumari, **Comparative study of software clone detection techniques,** In *2016 Management and Innovation Technology International Conference (MITicon)*, pp. MIT-152, *IEEE*, 2016

[36] O. Dahiya, and K. Solanki, **A systematic literature study of regression test case prioritization approaches,** *International Journal of Engineering & Technology*, vol. 7, no. 4, pp.2184-2191, 2018.

[37] K. Solanki, Y. Singh, and S. Dalal, **Experimental analysis of m-ACO technique for regression testing,** *Indian Journal of Science and Technology*, vol. 9, no. 30, pp.1-7, 2020.

[38] O. Dahiya, K. Solanki and S. dalal, **Comparative Analysis of Regression Test Case Prioritization Techniques,** *International Journal of advanced trends in computer science and engineering*, vol. 8 no. 4, pp. 1521-1531, 2019.

[39] A. Dhankhar and K. Solanki, **A Comprehensive Review of Tools & Techniques for Big Data Analytics,** *International journal of emerging trends in engineering research*, vol. 7, no. 11, pp. 556-562, 2019.

[40]M. Devi and N. S. Gill, **Mobile Ad Hoc Networks and Routing Protocols in IoT Enabled Smart Environment: A Review**, *Journal of Engineering & Applied Science*, vol. 14, issue 3,pp. 802-8011, 2019, DOI: 10.36478/jeasci.2019.802.811

[41]M. Devi and N. S. Gill, **Comparison Analysis of MANET Routing Protocols to identify their Suitability in Smart Environment,** *International Journal of Engineering and Technology (UAE)*, vol. 7, Issue 4, pp. 4844-4849,2018, DOI: 10.14419/ijet.v7i4.27945

[42]M. Devi and N. S. Gill, **Novel Algorithm for Enhancing Bitrate in MANET for Topology based routing protocol,** *International Journal of Engineering and Advanced Technology*, vol. 9, Issue 1, pp.2655-2662, 2019, DOI: 10.35940/ijeat.A9882.109119.

[43]D. Sehrawat, N. S. Gill and M. Devi, **Comparative Analysis of Lightweight Block Ciphers in IoT-Enabled Smart Environment,** *in: Proc. 2019 6th International Conference on Signal Processing and Integrated Networks (SPIN), IEEE*, pp. 915-920, 2019. Available: https://doi.org/10.1109/SPIN.2019.8711697

[44]M. Devi and N. S. Gill, **Performance Evaluation of Dynamic Source Routing Protocol in Smart**

Environment, *International Journal of Advanced Trends in Computer Science and Engineering*, vol.8, issue 2, pp. 333-338,2019,https://doi.org/10.30534/ijatcse/2019/37822019,

[45]M. Devi and N. S. Gill, **Novel Algorithm for Enhancing MANET Protocol in Smart Environment,** *International Journal of Innovative Technology and Exploring Engineering, vol. 8, issue 10, pp. 1830-1835, 2019,DOI: 10.35940/ijitee.J9214.0881019.*

[46]M. Devi and N. S. Gill, **Exploring Possibilities of MANET Protocols for IoT Enabled Smart Environment,** *International Journal of Computer Sciences and Engineering, vol. 7, issue 3, pp. 684-688, 2019, https://doi.org/10.26438/ijcse/v7i3.684688*

[47]M. Devi and N. S. Gill, **Study of Mobile Ad hoc Network Routing Protocols in Smart Environment**, *International Journal of Applied Engineering Research*, vol. 13, no. 16, pp. 12968-12975, 2018.

[48]M. Devi and N. S. Gill, **Performance Analysis of Enhanced Ad-Hoc On-Demand Distance Vector Routing Protocol in Smart Environment,** *International Journal of Recent Technology and Engineering*, vol. 8, issue 2, pp. 1548-1554, 2019, DOI: 10.35940/ijrte.B2227.078219

[49]M. Devi and N. S. Gill, **Challenges for Smart Environment: A Review**, *International Journal of Academic Research and Development.* vol. 3, issue 2, pp. 1277-1281, 2018.

[50]S. L. Ho, M. Xie and T. N. Goh, **A Study of the Connectionist Models for Software Reliability Prediction**, *Computers and Mathematics with Applications* , vol. 46, pp. 1037-1045, 2003.

[51]S. H. Aljahdali and K. A. Buragga, **Employing four ANNs Paradigms for Software Reliability Prediction: an Analytical Study**, *ICGST-AIML Journal*, ISSN: 1687-4846, vol. 8, issue II, 2008.

[52]M. Bisi and N. K. Goyal, **Software Reliability Prediction using Neural Network with Encoded Input**, *International Journal of Computer Applications (0975 – 8887)*, vol. 47, no. 22, pp. 46-52, 2012.

[53]S. K. Jain and M. P. Singh, **Estimation for Faults Prediction from Component Based Software Design using Feed Forward Neural Networks**, *IJARCCE*, vol. 2, issue 7, 2013.

[54]S. Chatterjee, S.Nigam, J.B.Singh, and L.N.Upadhyaya, **Application of Fuzzy Time Series in Prediction of Time Between Failures& Faults in Software Reliability Assessment**, *Fuzzy Information and Engineering*, vol. 3, no. 3, pp. 293-309, 2011.

[55]S. Aljahdali, **Development of Software Reliability Growth Models for Industrial Applications Using Fuzzy Logic**, *Journal of Computer Science*, vol. 7, no. 10, pp. 1574-1580, 2011.

[56]K. Khatatneh and T. Mustafa, **Software Reliability Modeling Using Soft Computing Technique**, *European Journal of Scientific Research, ISSN 1450-216X*, Vol.26 No.1, pp.147-152, 2009.

[57]E. O. Costa, A. T. R. Pozo, and S. R. Vergilio, **A Genetic Programming Approach for Software Reliability Modeling**, *IEEE Transactions on Reliability*, vol. 59, no. 1, 2010.

[58]Z. Al-Rahamneh, M. Reyalat, A. F. Sheta, SuliemanBani-Ahmad, S. Al-Oqeili, **A New Software Reliability Growth Model: Genetic-Programming-Based approach**, *Journal of Software Engineering and Applications*, vol. 4, pp. 476-481, 2011.

[59]R.S. Prasad., O. N. Raju and R. R. L. Kantam, **SRGM with Imperfect Debugging by Genetic Algorithms**, *International Journal of Software Engineering & Applications*, vol. 1, no. 2, pp. 66-79, 2010.

[60]Q. Yuexia and G. Weijie, **The Research on Reliability Optimization of Software System Based on Niche Genetic Algorithm**, *AASRI Conference on Computational Intelligence and Bioinformatics, AASRI Procedia 1*, pp. 404 – 409, 2012

[61]R. Satyaprasad, G. Bharathi, **Assessing Software Reliability Using Modified Genetic Algorithm: Inflection SShaped Model**, *International Journal on Future Revolution in Computer Science & Communication Engineering*, vol. 3, issue 11, pp. 136-141, 2017.

[62]L. Shanmugam and L. Florence, **Enhancement and comparison of ant colony optimization for software reliability models**, *Journal of Computer Science*, vol. 9, no. 9, pp. 1232-1240, 2013.

[63]R. Mohanthy, V. Naik, and A. Mubeen, **Software Reliability Prediction by Using Ant Colony Optimization Technique**, *Fourth International Conference on Communication Systems and Network Technologies*, pp. 496-500, 2014, doi:10.1109/csnt.2014.105

[64]Kiran Kumar B., J. Gyani and Narsimha G., **Software Defect Prediction using Ant Colony Optimization**, *International Journal of Applied Engineering Research*, vol. 13, no. 19, pp. 14291-14297, 2018

[65]D. HemaLatha, and P. Premchand, **Estimating Software Reliability Using Ant Colony Optimization Technique with Salesman Problem for Software Process,** *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 7, no. 2, pp. 20-29, 2018.

[66]H. Youssef, S. S. Sait, and H. Adiche, **Evolutionary algorithms, simulated annealing and tabu search: a comparative study,** *Engineering Applications of Artifical Intelligence*, vol. 14, issue 2, pp. 167-181, 2001. DOI: https://doi.org/10.1016/S0952-1976(00)00065-8

[67]M. Benaddy and M. Wakrim, **Simulated Annealing Neural Network for Software Failure Prediction,** *International Journal of Software Engineering and Its Applications*, vol. 6, no. 4, pp. 35-46, 2012.

[68]Y. K. Franty and B. Handoko, **The Determination of Preventive Maintenance using Simulated Annealing Algorithm Based on Weighted Fitness Function**, *JurnalTeknikIndustri*, vol. 20, no. 1, pp. 53-61, 2019

[69]K. Tyagi and A. Sharma, **An adaptive neuro fuzzy model for estimating the reliability of component-based software systems**, *Applied Computing and Informatics*, vol. 10, issues 1–2, pp. 38-51, 2014.

[70]Dhavakumar P, Shankar.S, and VikramPandi M, **Soft Computing Techniques For Enhancing Software Reliability**, *International Journal of Latest Trends in Engineering and Technology*, Special Issue, pp. 133-140, 2018.

[71]B. Kotaiah, M V S Prasad and R. A. Khan, **An Analysis of Software Reliability Assessment with Neuro- Fuzzy based Expert Systems**, *Procedia Computer Science*, vol. 62, pp. 92-98, 2015

[72]NajlaAkram AL-Saati and MarwaAbd-AlKareem, **The Use of Cuckoo Search in Estimating the Parameters of Software Reliability Growth Models**, *International Journal of Computer Science and Information Security*,vol. 11, no. 6, 2013.

[73]M. Dixit, P. Kundu and H. R. Jariwala, **Assessment of distribution system reliability using artificial bee colony algorithm,** *Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pp. 1-6, 2017.

[74]H. J. Li, J. J. Li and F. Kang, **Artificial Bee Colony Algorithm for Reliability Analysis of Engineering Structures,** *Advanced Materials Research*, vol. 163-167, pp. 3103-3109, 2010. doi:10.4028/www.scientific.net/amr.163-167.3103