Journal of Scientific & Industrial Research
Vol. 83, June 2024, pp. 617-626
DOI: 10.56042/jsir.v83i6.5415

Artificial Neural Network based Model for Reliability Assessment of Component based Software

Sumit Babu* & Raghuraj Singh

Department of Computer Science and Engineering, Harcourt Butler Technical University, Kanpur 208 002, Uttar Pradesh, India

Received 06 September 2023; revised 03 January 2024; accepted 13 May 2024

Software quality assessment during early phases of software development process is an extremely important concern of the researchers today because it identifies various aspects of quality degradation much prior to doing the actual damage to the final product quality. This also serves as the basis for improvement of the process for developing software. Software reliability is one of prime factor that affects software quality. In this paper, a model based on Artificial Neural Network (ANN) for the assessment of reliability of Component Based Software Systems (CBSS) has been proposed. First, a mathematical model based on formulation of software reliability in terms of the reliability factors using Analytical Hierarchy Process (AHP) is developed. Further, this model is refined by using ANN to calculate appropriate weight values reflecting true influence of reliability factors on software reliability. Model is validated by assessing the quality of 100 component based software using proposed models and an existing model. Results show a good correlation between the mathematical model and the ANN model. The proposed AHP model and ANN model achieve higher average reliability value of 0.5109 and 0.5088 respectively in comparison to the average reliability value of existing software reliability model (0.2261). Precise assessment of software reliability through the proposed models during early stages of software development helps developers to improve quality of software.

Keyword: Complexity, Fuzzy, Machine learning, Reusability, Software quality

Introduction

Due to our society's growing reliance on software and the frequently disastrous effects that a software failure can have in terms of lost lives, monetary losses, or delays. There is a growing demand for high-quality software. Every software system is expected to function consistently without failures. To determine effectiveness of software, its certain aspects must be measurable

Software quality refers to the level of customer satisfaction in relation to the fulfillment of their needs and expectations. Software quality depends on certain quality factors such as maintainability, functionality, understandability and most important the reliability. Various software quality models standards that describe sets of factors on which the software quality is dependent discussed by Babu and Singh.[1] Software reliability is still a vague and complicated concept, with several interpretations and can described as "the probability that software will not cause breakdown of a system for a set time under a specific operational environment".[2]

In today's world, where technologies are updating regularly, the Component Based Software Development (CBSD) technique is a more contemporary method that is built on aggregating individual components into software systems of complex nature. This technique deals with the usage of software components in development process which reduces the efforts, cost, time and complexity[3] and offers a number of benefits, including software quality, component reusability, productivity, lower maintenance costs, and faster time to market the product. Advantages of the CBSD approach are depicted in the Fig. 1.

AHP is a well-known technique of assigning relative weights or creating rational scales for the components of a system using paired discrete or continuous comparisons. Changes in measurement, relationships within and between groups of structural components, and deviations from consistency are of special importance to the AHP. Its uses in planning, multi-criteria decision making and conflict resolution have been the most widespread. In order to use the AHP and perform pair-wise comparisons to build relationships within the structure, the problem must be represented as a hierarchical or network structure.[4]

*Author for Correspondence
E-mail: 180304008@hbtu.ac.in

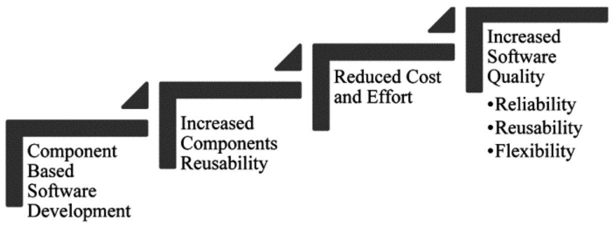

Fig. 1 — Advantages of CBSD

In this study, mathematical model for prediction of reliability of software developed on CBSD approach using AHP has been proposed. The model has been further refined by using ANN. Performance of both the models has been compared with the model proposed by Diwaker *et al*.[5]

Literature Review

The literature in software reliability modeling is extensive and vast. This section described the recent research consisting of unification or integration of the basic models in general and for the CBSS in particular.

Diwaker *et al*.[5] proposed reliability prediction models using three techniques for CBSS. Prediction model used new mathematical approach that employs series and parallel reliability models, fuzzy logic and Particle Swarm Optimization (PSO) techniques. The results demonstrate that fuzzy logic is superior to PSO for predicting reliability. In comparison to reliability prediction using fuzzy logic and PSO, the mathematical model has low error rate. But, still there is scope of using other techniques in place of fuzzy and PSO for the development of hybrid approach.

Models for the growth of software dependability are basically statistical interpolations of the data from mathematical models for fault detection. With less normality assumptions, Thakur and Sharma[6] found the optimum confidence interval in their study. The most popular software development processes were examined in Yakovyana *et al*.[7] and 15 key criteria that influence the quality and reliability of software were found. The elements of the first stage of the life cycle have been identified as having the greatest impact on the quality and reliability of the final product.

Jagtap *et al*.[8] proposed a model in which software metrics based on development data are used to calculate software reliability, which is a function of the amount of residual failures. It was discovered that combining a pattern recognition algorithm approach for categorizing fault proneness with applying fuzzy logic to software measurements for defect prediction can improve software reliability prediction in the early stages of development. Zhang *et al*.[9] also proposed a model for reliability which uses software metrics. In this approach they pointed out the redundant metrics and aggregated other metrics to find reliability. However, both the models had the limitation of selecting optimal set of metrics for the reliability assessment.

An entropy-combinative distance-based assessment method was presented by Garg *et al*. [10] to select and rank software reliability growth model based on several performance measures. Lin and Huang[11] proposed a method to use queuing based implementation to explain the behavior of fault correction process and assessment of software reliability. The approach used by these models is purely conventional and is not based on integration of the recent approaches.

Wu and Huang[12] proposed a reliability prediction model which uses deep learning approach. They demonstrated a method to establish relationship between the mathematical expressions developed using deep learning based computational techniques and the software reliability growth model's formula. Awasthi and Sharma[13] presented a study of various software reliability models using NN. Although these models used recent approaches for the development of reliability expression but they developed the expression on the basis of reliability growth modeling. However, it is always better if the reliability expression is developed using parameter based reliability assessment. Zhen *et al*.[14] proposed a hybrid method of wolf pack algorithm and PSO and used probabilistic estimation to generate a fitness function. The limitation of these hybrid approaches are that they did not consider different pairs of the optimization strategies to determine the best pair to be used for development of software reliability assessment model.

A simulation-relied framework created by Diwaker and Tomar[15] enables a thorough study of fault injection on hypervisors with a variety of configurations. Many hardware problems have been known to broadcast over various paths for a long period before being noticed. Jaiswal and Giri[16] used Fuzzy Inference System (FIS) and Adaptive Neuro-Fuzzy Inference System (ANFIS) with two different membership functions to assess CBSS dependability. Reusability, component dependency, operational profile and application complexity were four aspects that were taken into consideration as parameters. An ANFIS model was proposed by Tyagi and Sharma[17] for assessing CBSS reliability with various statistics sets. This hybrid strategy needed less computation time. FIS was outperformed by ANFIS. For large data sets, the model executed sophisticated operations.

Heuristic Component Dependency Graphs (HCDGs) were presented by Tyagi and Sharma[18] to estimate CBSS reliability, including component reliability. Identification of most popular path for the estimation of CBSS reliability is a good idea but determining weighing factors to these paths on the basis of their relative criticality and considering pheromone dissipation also as a parameter will further improve the accuracy of the results. A technique to evaluating dynamic software performance that takes the consequences of soft faults into account was put forth by Diwaker and Tomar[19] a model was used that combined abstract calculation on a high level with low level calculation instructions. The dynamic program reliability model is verified by the results of fault injection testing. A model put forth by Tyagi and Sharma[20] focused on 4 elements that have a significant impact on CBSS reliability. The method for calculating CBSS dependability utilizes fuzzy logic. Reusability, operation profile, complexity, and component dependency made up these criteria. Future work may also be influenced by other circumstances.

When estimating the reliability of the components, Goswami *et al.*[21] Fuzzy-Inference model provides a clear response for business personnel. Statistical approaches, soft computing techniques, inspections, measurement design, and inspections are used to predict early reliability. Wang *et al.*[22] proposed a reliability model based on the G-O model considering two parameters namely the total number of system faults and the rate at which those problems are found. By analyzing the relationship between these values, the defined system parameters are derived from the known component parameters. ChauPattnaik *et al.*[23] proposed a fuzzy set based reliability estimation model. They incorporated features failure rate, transition probability and component reliability. However, the model has a major drawback that it does not consider component's failure probability.

From the literature review, it may be concluded that although enough work on reliability assessment of CBSS has been done but still there is a need to refine various existing reliability assessment models using suitable reliability factors and metrics along with the advance techniques like ANN, machine learning (ML), fuzzy logic and other soft computing based techniques for better aggregation and refinement. In this study five major aspect of CBSS reliability are considered for reliability assessment. Relative strength of each factor is found using AHP to achieve higher reliability in comparison to various existing models.

Proposed Models for Software Reliability Assessment

The majority of software reliability models that are used for reliability assessment predict reliability during the testing and operation phases of the product after product development are complete. These models hardly have any influence on the planning and design phases of the software project because they are based on data produced after development. Once errors have caused damage, these models intervene to assess the damage and, at most, make recommendations for the necessary dosages of corrective action. These models offer information that is not timely enough to enhance internal product attributes before the product is finished. Hence, it is important to create models that may be utilized during the first phases of development (requirements and design) to guarantee that the analysis and design possess beneficial internal characteristics, ultimately resulting in the creation of a reliable final product. Developers would have the ability to resolve issues, rectify anomalies and noncompliance with standards, and minimize unnecessary complexity at an early stage of the development process.

The model presented here can be used for the assessment of reliability of CBSS during the initial phases of development, that is, at the time of requirement specifications and design. The model utilizes values of high level design factors obtained through low-level design metrics to assess the reliability through an aggregation process. First, we propose a mathematical model for the aggregation of various factors with aggregation process achieved through the AHP by working out the relative influence of the individual factors on the reliability of overall software system. The method is further refined by using ANN to determine weights/influence of the individual components and assess the refined reliability value.

Reliability Factors

Factors like reusability, interaction, dependency and failure are of prime concern for the development of a new software product using CBSD approach. The reliability of CBSD depends on the component's ability to be reused with minimal changes to develop new software product which can fulfill customer requirements.[24] By estimating the reliability of each component separately and the connectivity methodology between components, the reliability may be predicted.[25] A large number of models including architecture-based models, the Everett's model[26], Yacoub's model[24], Gokhale's model[27], Shooman's model[28], Laprie model[29] and many others exist to predict reliability of software systems. The parameters frequently used in these models include availability, failure rate, failure behavior of components and interfaces, operation profile, arithmetic algorithm errors, component reliability, transition probabilities, Component Dependency (CD), Component Interaction (CI), number of faults, mean repair times, execution of a set of components, series and parallel component combinations used, average component execution time etc.[5] However, the main and important factors used for the assessment of reliability in CBSD are Reusability, CI, CD, Complexity and Failure are the main elements to influence reliability of CBSS. For a certain type of problem, the priority and ranking of these factors may vary.

1. **Reusability (R):** Reusability is the ability to reuse the components of software. These components include program logic, a loop, certain proportion of line of code and a function of several classes that can be used to create new software. Understandability, maintainability, flexibility, variability, portability are the sub-parameters of reusability.[5] As shown in Jaiswal and Giri[16], there is a correlation between component reliability and reusability.

Reliability $\propto$ Reusability

2. **Component Interaction (CI):** The interaction highlights the interconnected components at the interface. Because of this, components become more reusable. As a result, overall reliability will be increased. Component interaction and reliability can be described as follows.[20]

Reliability $\propto$ Component Interaction

3. **Component Dependency (CD):** Dependency on other components is referred to as component dependency. Increased dependability indicates a more combined perspective of the components. Low reliability is caused by more component dependability. Component dependency and reliability can be described as follows.[15]

Reliability $\propto$ (1/Component Dependency)

4. **Complexity (C):** The amount of interconnections between components, the time it takes to execute an instruction, how much memory is required and the type of platform are all factors that affect complexity. It is directly proportional to number of components. High complexity software results in low reliability.[20] The relationship between complexity and reliability can be stated as follows in Tyagi and Sharma[18]:

Reliability $\propto$ (1/Complexity)

5. **Failure (F):** It is the frequency with which a component or system fails. It covers availability, mean time to repair, mean time before repair, and mean time to failure. Low reliability is caused by more system failure. Reliability and failure have the following relationships[18,22]:

Reliability $\propto$ (1/Failure)

Mathematical Model

For the development of mathematical model, we consider Reusability (R), CI, CD, Complexity (C) and Failure (F) factors as input parameters to determine the CBSS reliability using the Eq. (1).

$$\textbf{Reliability} = W1 * R + W2 * CI + W3 * CD + W4 * C + W5 * F \quad \dots (1)$$

where, weights W_1, W_2, W_3, W_4, and W_5 are determined using AHP.

AHP is based on creation of hierarchical or network structure showing relationship among various factors, criteria, characteristics or metrics. For determination of various weights, the hierarchical structure is shown in Fig. 2.

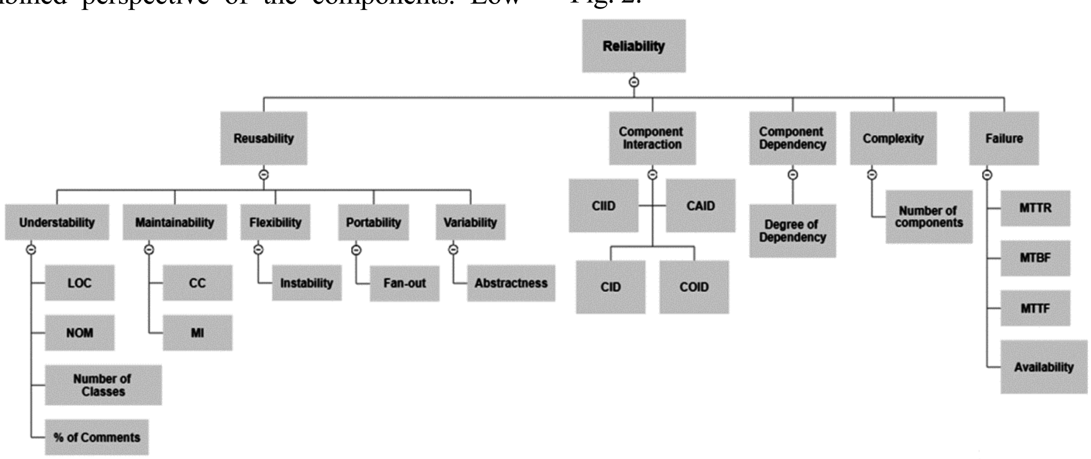

Fig. 2 — Hierarchical structure for reliability

The hierarchy shows further dependence of various reliability factors on certain other measures. For example, Reusability depends on variability, flexibility, portability, understandability and maintainability. Understandability further depends on the Line of Code (LOC), Number of Method (NOM), Number of Classes (NOC) and Percentage of Comments. Maintainability depends on Cyclomatic Complexity (CC) and Method Interaction (MI) metrics. Flexibility depends on the Instability. Portability depends on the Fan-out metric and variability depends on the Abstractness of software. Component Interaction depends on the Component interaction Density metrics such as CIID (Component Incoming Interaction Density), CAID (Component Average Interaction Density), CID (Component Interaction Density) and COID (Component Outgoing Interaction Density). Component dependency depends on the Degree of Dependency of components. Complexity depends on the number of components and Failure includes MTTR (Mean Time to Repair), MTTF (Mean Time to Before Failure), MTTF (Mean Time to Failure) and Availability.

In step 2, the pair-wise comparison is done with expert opinion to build pair-wise matrix and normalization of these values is done within the range 0 to 1.

In step 3 weights of relative components are found and other values such as $Lamda_{max}$, Consistency Index and Random Consistency Index (RI) are calculated for the problem. If the value of RI is less than 1.12 for five parameters then the weights are acceptable[4].

For the aggregation process, first the metric values of software are found using various metric tools. Then these values are used to find the values of sub-factors which are further used to find the values of reliability factors. Lastly CBSS reliability is found using the equation (1). After applying AHP to CBSD approach we find weights as W_1= 0.5978, W_2= 0.1312, W_3= 0.1228, W_4= 0.0809 and W_5= 0.0673.

The schematic description of the AHP followed for determination of various weights is represented in Fig. 3.

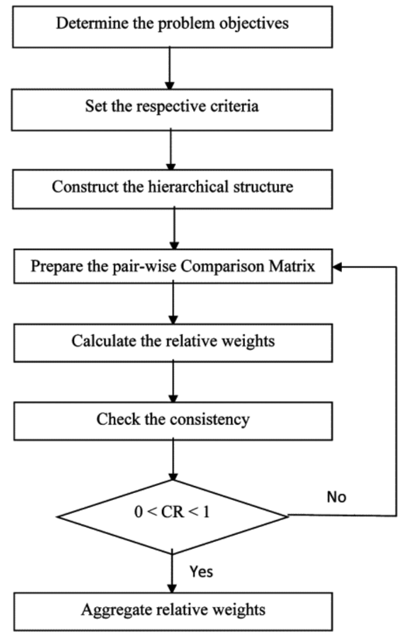

Fig. 3 — Flow chart of AHP process

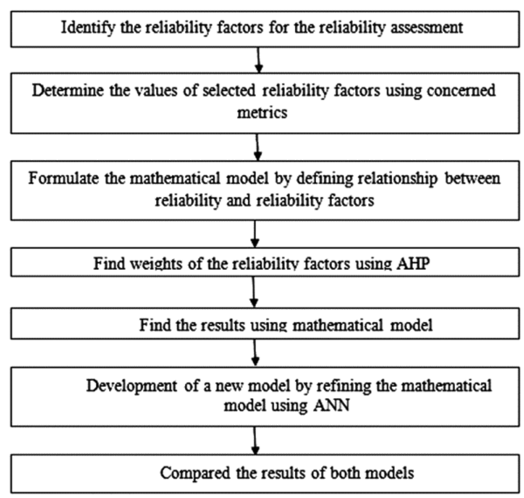

Fig. 4 — Flow chart for ANN model development

ANN based Model

To further refine the values of various weights assigned to different reliability factors used for the assessment of reliability of CBSS, we propose an ANN based reliability assessment model. ANN is made up of artificial neurons having their working similar to the biological neurons and it is a set of vertical components called layers. Input, hidden, and output layers make up the majority of its layers. The neurons receive input at the input layer, where it is enhanced by the addition of bias and weight. While bias is added to inputs to alter them inside neurons, weight indicates how important the input is. The performance and complexity of the network are handled by the hidden layer, which may consist of several layers depending on the kind and requirements of the network. The input for the output layer with weight and bias is the output of hidden layers. The steps for ANN model development are shown in Fig. 4.

Here, we use ANN to aggregate various reliability factors by considering their appropriate weights finalized by the single hidden layer neural network as shown in Fig. 5. The values of five reliability factors found through respective metrics are treated as inputs for the neural network.

Model Implementation

The values of reliability factors of 100 component based programs of low, medium and high design complexity taken from Diwaker *et al.*[5] are shown in Table 1. Results of the reliability assessment through an existing model, proposed mathematical model using AHP as well as proposed ANN based model are also given in the Table 1. The values of reliability factors namely reusability, component interaction, component dependency, complexity and failure lie between 0 and 1. The values from 0 to 0.34, 0.35 to 0.68 and 0.69 to 1 indicate low, medium and high level designs of the programs respectively. The final values of reliability assessed through various models also lie between 0 and 1. During different epochs of the ANN, the weights and bias are automatically adjusted. The dataset is split into two sets, a training set and a testing set in the ratio of 80 and 20%.The model is simulated using python programming language.

Results and Discussion

The reliability values of various software are shown in the Table 1 using proposed AHP based mathematical model. Correlation or dependence which is any

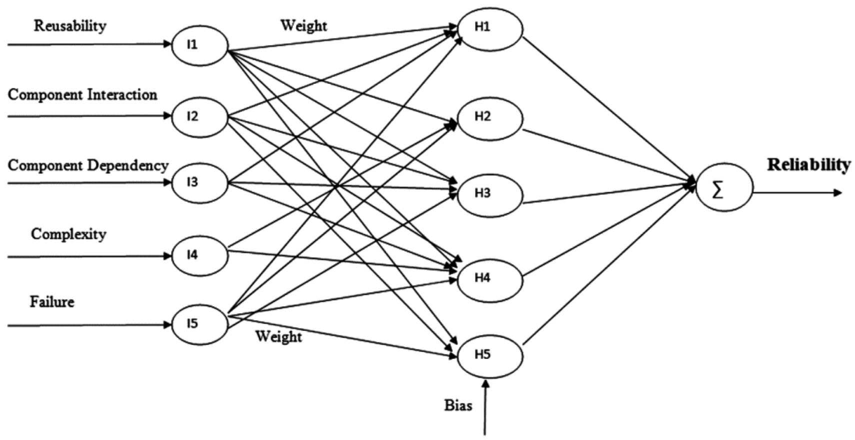

Fig. 5 — Structure of proposed neural network

Table 1 — Values of reliability factors and Reliability using various models

S. No.	Reusability	CI	CD	Complexity	Failure	Reliability using Existing Model	Reliability using AHP	Reliability using ANN
1	0.8147	0.6557	0.4387	0.7513	0.3517	0.4858	0.7113	0.7084
2	0.189	0.509	0.1155	0.4791	0.8819	0.0909	0.2918	0.2906
3	0.6081	0.5141	0.5088	0.4448	0.6713	0.2846	0.5744	0.5721
4	0.5058	0.8634	0.1371	0.7478	0.0704	0.3483	0.4977	0.4957
5	0.7284	0.642	0.7584	0.1684	0.7327	0.4425	0.6755	0.6728
6	0.6905	0.9525	0.1552	0.6118	0.1052	0.4646	0.6134	0.6109
7	0.4677	0.8105	0.501	0.6931	0.9734	0.3775	0.5687	0.5665
8	0.6237	0.0257	0.0207	0.5744	0.3994	0.012	0.4520	0.4502
9	0.4195	0.6464	0.0911	0.2161	0.5607	0.1862	0.4018	0.4002
10	0.6936	0.1511	0.9969	0.8395	0.1492	0.1047	0.6348	0.6323
11	0.3239	0.4396	0.9453	0.7069	0.2121	0.1405	0.4388	0.4370
12	0.3528	0.6885	0.4216	0.777	0.223	0.2185	0.4308	0.4291
13	0.9915	0.64	0.7555	0.9126	0.7708	0.6314	0.8949	0.8914
14	0.2124	0.7348	0.0001	0.678	0.5295	0.1324	0.3137	0.3125
15	0.2284	0.2356	0.5857	0.4664	0.6827	0.05	0.3228	0.3216
16	0.9464	0.7616	0.4357	0.3999	0.6089	0.6253	0.7923	0.7892
17	0.4699	0.1837	0.2462	0.5463	0.3692	0.0676	0.4042	0.4026
18	0.5995	0.6941	0.6739	0.7973	0.9306	0.4142	0.6591	0.6564
19	0.9834	0.7493	0.2526	0.391	0.2142	0.4733	0.7632	0.7601
20	0.8821	0.2123	0.4829	0.3154	0.2641	0.1384	0.6577	0.6551

(*Contd.*)

Table 1 — Values of reliability factors and Reliability using various models *(Contd.)*

S. No.	Reusability	CI	CD	Complexity	Failure	Reliability using Existing Model	Reliability using AHP	Reliability using ANN
21	0.7974	0.9072	0.1148	0.9989	0.4634	0.723	0.7217	0.7188
22	0.5394	0.6486	0.3594	0.976	0.5072	0.3472	0.5646	0.5624
23	0.0745	0.1609	0.1486	0.9368	0.9091	0.0119	0.2206	0.2197
24	0.4442	0.7012	0.7968	0.0631	0.8369	0.3018	0.5166	0.5145
25	0.8466	0.1165	0.001	0.9603	0.907	0.0982	0.6600	0.6573
26	0.9075	0.8527	0.1324	0.3519	0.7239	0.0702	0.7476	0.7446
27	0.0306	0.939	0.7205	0.331	0.9741	0.0285	0.3220	0.3207
28	0.1075	0.442	0.5373	0.3191	0.554	0.0408	0.2512	0.2502
29	0.9606	0.512	0.6475	0.0674	0.8791	0.4722	0.7853	0.7821
30	0.9463	0.7616	0.155	0.9089	0.1803	0.6752	0.7703	0.7672
31	0.1949	0.5132	0.6588	0.2105	0.7359	0.0929	0.3311	0.3298
32	0.1178	0.0077	0.4815	0.3358	0.8656	0.0008	0.2157	0.2149
33	0.3109	0.7797	0.2933	0.2048	0.0767	0.1166	0.3459	0.3445
34	0.9808	0.5503	0.0031	0.0635	0.5721	0.3241	0.7024	0.6996
35	0.5539	0.19	0.7395	0.7173	0.3528	0.0974	0.5285	0.5264
36	0.5604	0.2156	0.6414	0.2039	0.3223	0.0974	0.4801	0.4782
37	0.2359	0.0618	0.3583	0.5866	0.3977	0.0122	0.2672	0.2662
38	0.1307	0.6207	0.6642	0.3569	0.8847	0.0791	0.3293	0.3280
39	0.7417	0.9589	0.5829	0.0436	0.107	0.4577	0.6515	0.6489
40	0.1052	0.747	0.6172	0.5426	0.4125	0.0705	0.3082	0.3070
41	0.2607	0.1927	0.9911	0.5081	0.3912	0.0501	0.3702	0.3687
42	0.882	0.2659	0.5928	0.2388	0.1513	0.1728	0.6644	0.6617
43	0.8428	0.7447	0.2251	0.7696	0.9289	0.6196	0.7537	0.7507
44	0.2128	0.0569	0.5891	0.0727	0.9679	0.0119	0.2777	0.2766
45	0.723	0.4867	0.8943	0.8646	0.4737	0.3492	0.7076	0.7047
46	0.7109	0.6716	0.1092	0.8855	0.9517	0.475	0.6619	0.6593
47	0.4516	0.9458	0.4151	0.0528	0.6443	0.3429	0.4925	0.4905
48	0.0974	0.6658	0.0795	0.0147	0.7868	0.0523	0.2092	0.2084
49	0.3538	0.604	0.3337	0.0482	0.5407	0.1514	0.3719	0.3704
50	0.9887	0.3171	0.5773	0.2432	0.767	0.2902	0.7746	0.7715
51	0.0959	0.335	0.651	0.7255	0.1	0.0296	0.2466	0.2456
52	0.7977	0.5856	0.4265	0.8726	0.9852	0.4666	0.7427	0.7397
53	0.1515	0.6203	3747	0.0676	0.9778	0.092	0.2889	0.2878
54	0.4701	0.0078	0.2735	0.3835	0.3124	0.0025	0.3676	0.3661
55	0.4852	0.1577	0.1055	0.7722	0.5262	0.0691	0.4214	0.4197
56	0.1832	0.2626	0.0466	0.4608	0.7548	0.0424	0.2375	0.2366
57	0.3291	0.2351	0.518	0.9622	0.3231	0.0764	0.3907	0.3891
58	0.7429	0.0032	0.7707	0.6562	0.6402	0.0023	0.6351	0.6326
59	0.7541	0.8895	0.1257	0.0366	0.0864	0.1545	0.5917	0.5893
60	0.0849	0.0152	0.85	0.3416	0.8686	0.0012	0.2430	0.2420
61	0.8779	0.8867	0.2372	0.6339	0.4057	0.6492	0.7487	0.7457
62	0.0633	0.5593	0.1969	0.5969	0.9211	0.0344	0.2454	0.2444
63	0.9009	0.1066	0.762	0.1085	0.0839	0.0773	0.6605	0.6579
64	0.4891	0.368	0.0092	0.1123	0.6446	0.1237	0.3941	0.3925
65	0.1919	0.032	0.9976	0.908	0.1276	0.0061	0.3234	0.3221
66	0.3223	0.36	0.9222	0.3927	0.9466	0.1157	0.4483	0.4465
67	0.3502	0.2277	0.9651	0.8163	0.1976	0.0793	0.4370	0.4353
68	0.135	0.3788	0.2458	0.6863	0.9823	0.0509	0.2819	0.2808
69	0.7629	0.9522	0.191	0.0779	0.7692	0.6013	0.6623	0.6596
70	0.4314	0.2158	0.4634	0.3942	0.5823	0.0824	0.4140	0.4124
71	0.6818	0.9818	0.8208	0.1527	0.7393	0.6428	0.6991	0.6963

(*Contd.*)

Table 1 — Values of reliability factors and Reliability using various models *(Contd.)*

S. No.	Reusability	CI	CD	Complexity	Failure	Reliability using Existing Model	Reliability using AHP	Reliability using ANN
72	0.8675	0.0525	0.2342	0.6152	0.8041	0.0429	0.6579	0.6553
73	0.7048	0.3881	0.3972	0.6182	0.9309	0.2691	0.6334	0.6309
74	0.2318	0.7803	0.9705	0.9212	0.5675	0.1806	0.4727	0.4708
75	0.1246	0.7492	0.485	0.9771	0.877	0.0932	0.3701	0.3687
76	0.7717	0.6402	0.5865	0.6365	0.5893	0.4635	0.7083	0.7055
77	0.6845	0.6133	0.4895	0.0961	0.8717	0.3948	0.6159	0.6135
78	0.2644	0.5731	0.7149	0.1174	0.1697	0.1198	0.3419	0.3405
79	0.3229	0.1278	0.3173	0.8216	0.8641	0.0405	0.3731	0.3716
80	0.4662	0.906	0.352	0.9463	0.2767	0.4117	0.5359	0.5337
81	0.2584	0.6878	0.7014	0.0936	0.86	0.1709	0.3960	0.3945
82	0.8022	0.0296	0.6127	0.0435	0.2556	0.0171	0.5793	0.5770
83	0.8965	0.2402	0.2876	0.4741	0.5914	0.1823	0.6807	0.6780
84	0.6199	0.912	0.3584	0.7367	0.4767	0.5153	0.6258	0.6233
85	0.4252	0.6657	0.8748	0.996	0.9096	0.283	0.5905	0.5881
86	0.8012	0.7849	0.4393	0.683	0.7797	0.6042	0.7434	0.7404
87	0.8296	0.5981	0.7258	0.4447	0.2208	0.4373	0.7143	0.7114
88	0.0555	0.3625	0.8173	0.7177	0.6735	0.0197	0.2843	0.2832
89	0.7569	0.485	0.2821	0.0378	0.1868	0.1608	0.5663	0.5641
90	0.3746	0.1397	0.9072	0.1737	0.1798	0.049	0.3798	0.3782
91	0.9961	0.0688	0.8113	0.7567	0.6038	0.0672	0.8058	0.8026
92	0.4671	0.8673	0.2048	0.5254	0.4835	0.3261	0.4931	0.4911
93	0.2525	0.0981	0.4009	0.4585	0.0553	0.0171	0.2538	0.2528
94	0.7609	0.3628	0.3639	0.7552	0.7731	0.2663	0.6600	0.6574
95	0.9569	0.6837	0.435	0.5225	0.973	0.6494	0.8226	0.8193
96	0.341	0.1587	0.6916	0.3784	0.3969	0.0478	0.3668	0.3653
97	0.2386	0.7311	0.2673	0.7397	0.0102	0.1415	0.3319	0.3306
98	0.2716	0.7822	0.2465	0.6907	0.9941	0.2121	0.4177	0.4161
99	0.8133	0.8061	0.5199	0.2541	0.8904	0.6298	0.7360	0.7331
100	0.2192	0.3599	0.5834	0.61	0.8071	0.0764	0.3533	0.3519

statistical relationship or a type of association between two random variables, usually refers to the degree to which a pair of variables are linearly related. The correlation between the reliability values assessed through proposed mathematical model and ANN model is 0.9960 which represents a strong correlation between the two models. The proposed models achieve higher reliability as compared to the existing model. Particularly, the proposed AHP model and ANN model achieve higher average reliability value of 0.5109 and 0.5088 respectively in comparison to the average reliability value of the existing software reliability model (0.2261).

We used Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to evaluate the accuracy. The difference between the value predicted by a model and the observed values is measured using RMSE. It is the average of the squares of all the error square roots. Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to significant errors. In situations when significant errors are most undesirable, the RMSE is therefore most helpful. Without taking into account the direction of the differences, the MAE calculates the average of the forecasting errors. MAE assumes that all individual differences are equally weighted in the average because it is a linear score. Table 2 shows the value of RMSE and MAE for the ANN model. The R square value of the proposed mathematical model, which represents the coefficient of determination, is 0.9960 which is near to 1. It demonstrates that it is an effective predictor of reliability.

Table 2 — Error Values

Model	RMSE	MAE
ANN based reliability assessment	0.0106	0.0057

Conclusions

CBSD approach provides a way to reuse its components to reduce cost and efforts of software development process. The relative weighting of the reliability factors in the evaluation of software reliability has a significant impact. Precise assignment of relative weights to five selected factors namely Reusability, CI, CD, Complexity and Failure as well as various other criteria and metrics has been done using AHP and ANN. As a result, the prediction accuracy of CBSS reliability is enhanced. Results of mathematical model are compared with the results of the ANN model and it shows a strong correlation and low error. Also, both the proposed models achieve higher reliability values as compared to the existing models. Our work has the limitation that it assesses reliability assuming absolute transition among various components of CBSS. However, it will be more appropriate if actual transition probability among components is considered. In future, other combination of factors and metrics which affect reliability of software systems may be considered and more sophisticated multi-level, multi-criteria based methods like fuzzy-AHP ranking method may be used to assign relative weights and perform the aggregation process to assess the reliability.

References

1 Babu S & Singh R, Neural network-based model for the quality assessment of object-oriented software, *Int J Open Source Softw Process*, **13(1)** (2022) 1–13, https://doi.org/10.4018/ IJOSSP.313182.

2 Aggarwal K K, Reliability fundamentals, in *Reliability Engineering* (Springer, Dordrecht) 1993, 1–29.

3 Aloysius A & Maheswaran K, A review on component based software metrics, *Intern J Fuzzy Math Archive*, **7(2)** (2015) 185–194.

4 Saaty T L, Exploring the interface between hierarchies, multiple objectives and fuzzy sets, *Fuzzy Sets Syst*, **1(1)** (1978) 57–68, https://doi.org/10.1016/0165-0114(78) 90032-5.

5 Diwaker C, Tomar P, Solanki A, Nayyar A, Jhanjhi N Z, Abdullah A & Subramaniam M, A new model for predicting component-based software reliability using soft computing, *IEEE Access*, **7** (2019) 147191–147203, https://doi.org/ 10.1109/ACCESS.2019.2946862.

6 Thakur P & Sharma S K, Estimation of complexity in software reliability growth modeling, *Adv Appl Math Sci*, **19(6)** (2020) 563−572.

7 Yakovyna V, Seniv M & Symets I, The relation between software development methodologies and factors affecting software reliability, *IEEE 15th Int Conf Comput Sci Inform Technol (CSIT),* (2020) 37–381, https://doi.org/10.1109/ CSIT49958.2020.9321937.

8 Jagtap M, Katragadda P & Satelkar P, Software reliability, development of software defect prediction models using advanced techniques, *Ann Reliab Maintainab Sympos,* (2022) 1–7, https://doi.org/10.1109/ RAMS51457.2022. 9893986.

9 Zhang J, Lu Y, Shi K & Xu C, Empirical research on the application of a structure-based software reliability model, *IEEE/CAA J Automatica Sinica*, **8(6)** (2021) 1153–1162, https://doi.org/10.1109/JAS.2020.1003309.

10 Garg R, Raheja S & Garg R K, Decision support system for optimal selection of software reliability growth models using a hybrid approach, *IEEE Trans Rel*, **71(1)** (2022) 149–161, https://doi.org/10.1109/TR.2021.3104232.

11 Lin J S & Huang C Y, Queuing-based simulation for software reliability analysis, *IEEE Access*, **10** (2022) 107729−107747, https://doi.org/10.1109/ACCESS.2022.3213271.

12 Wu C Y & Huang C Y, A study of incorporation of deep learning into software reliability modeling and assessment, *IEEE Trans Rel*, **70(4)** (2021) 1621–1640, https://doi.org/ 10.1109/TR.2021.3105531.

13 Awasthi V & Sharma S K, A study of various software reliability systems by using ann, *J Univ Shanghai Sci Technol*, **23(7)** (2021).

14 Zhen L, Liu Y, Dongsheng W & Wei Z, Parameter estimation of software reliability model and prediction based on hybrid wolf pack algorithm and particle swarm optimization, *IEEE Access*, **8** (2022) 29354–29369, https://doi.org/ 10.1109/ACCESS. 2020.2972826.

15 Diwake C & Tomar P, Evaluation of swarm optimization techniques using CBSE reusability metrics, *Int J Control Theory Appl*, **2** (2016) 189–197.

16 Jaiswal G P & Giri R N, Software reliability estimation of component based software system using fuzzy logic, *Int J Comput Appl*, **127(7)** (2015) 16–20.

17 Tyagi K & Sharma A, An adaptive neuro fuzzy model for estimating the reliability of component-based software systems, *Appl Comput Inform*, **10(1–2)** (2014) 38–51, https://doi.org/10.1016/j.aci.2014.04.002.

18 Tyagi K & Sharma A, A heuristic model for estimating component-based software system reliability using ant colony optimization, *World Appl Sci J*, **31(11)** (2014) 1983–1991, https://doi.org/10.5829/idosi.wasj.2014.31.11.1731.

19 Diwaker C & Tomar P, Identification of factors and techniques to design and develop component based reliability model, *Int J Sci Res Comput Sci Eng*, **5(3)** (2017) 107–114.

20 Tyagi K & Sharma A, A rule-based approach for estimating the reliability of component-based systems, *Adv Eng Softw*, **54** (2012) 24–29, https://doi.org/10.1016/j.advengsoft. 2012. 08.001.

21 Goswami P, Noorwali A, Kumar A, Khan M Z, Srivastava P & Batra S, Appraising early reliability of a software component using fuzzy inference, *Electronics*, **12(5)** (2023) 1137, https://doi.org/10.3390/electronics12051137.

22 Wang Y, Liu H, Yuan H & Zhang Z, Comprehensive evaluation of software system reliability based on component-based generalized GO models, *Peer J Comput Sci*, **9** (2023) e1247, https://doi.org/10.7717/peerj-cs.1247.

23 ChauPattnaik S, Ray M & Nayak M, Fuzzy set-based reliability estimation, *Int J Softw Innov*, **11(1)** (2023) 1–14, https://doi.org/10.4018/IJSI.315733.

24 Yacoub S, Cukic B & Ammar H H, A scenario-based reliability analysis approach for component-based software, *IEEE Trans Rel*, **53(4)** (2004) 465–480, https://doi.org/ 10.1109/ TR. 2004.838034.

25 Goseva-Popstojanova K, Mathur A P & Trivedi K S, Comparison of architecture-based software reliability models, *in Proc 12th IEEE Int Symp Softw Rel* (IEEE) 2001, 22–31, https://doi.org/10.1109/ISSRE.2001.989455.

26 Everett W W, Software component reliability analysis, *Proc 1999 IEEE Symp Appl-Specific Syst Software Eng Technol* (Cat. No. PR00122) 1999, 204–211, https://doi.org/ 10. 1109/ASSET.1999.756770.

27 Gokhale S S & Trivedi K S, Analytical models for architecture-based software reliability prediction: A unification framework, *IEEE Trans Rel*, **55(4)** (2006) 578–590, https://doi.org/ 10.1109/TR.2006.884587.

28 Shooman M L, Structural models for software reliability prediction, *Proc 2nd IEEE Int Conf Softw Eng* (IEEE) 1976, 268–280.

29 Costes A, Landrault C & Laprie J C, Reliability and availability models for maintained systems featuring hardware failures and design faults, *IEEE Transact Comput*, **27(6)** (1978) 548–560, https://doi.org/10.1109/ TC.1978.1675146.