Received: August 23, 2018

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

The Robust Architecture Based Reliability Analysis Framework of Complex Software System Using In-Degree and Out-Degree

Kaliraj Shanmugaiah [1]* Bharathi Ayyaswamy [2]

[1]*Anna University, Chennai, India*
[2]*Bannari Amman Institute of Technology, Tamilnadu, India*
* Corresponding author's Email: kaliraj.se@gmail.com

Abstract: Nowadays, Software development process has incorporated many techniques, scientific approaches and advanced tools to develop the highest quality featured software. The size of the software systems also become large and make the system more complex. So that, it is necessary to ensure the quality of software system before its delivery. Software reliability is one of the important measurement to predict the quality of software which will be used to reduce the time, cost and other resources required for the testing process. Here we have proposed one novel framework to predict the software reliability based on software architecture and path testing by incorporating in-degree and out-degree concept used in graph theory. This proposed framework will reduce the number of test path in the system which will be used to reduce the testing time and resources. The applicability of our proposed framework is validated by using two real time case studies and these results are compared with the prevailing standard reliability models to assess the prediction accuracy of our proposed framework. The result shows that our proposed framework is simple to apply in the real time system and prediction accuracy is also acceptable compared to the other models.

Keywords: Architecture quality, Software reliability analysis, Software performance, Software quality prediction.

1. Introduction

All Software Reliability is the probability of failure-free software operation for a specified period of time in a specified environment. Early measurement of software reliability is an important factor in the software development process. It will be used to reduce the time, cost and resources required for the testing process. Additionally, software engineers and customers will get the confidence about the software product which is to be developed.

In the last two decades, many reliability methods have been proposed to estimate the reliability of software systems. But with the increasing size and complexity of software applications, the traditional software reliability methods are insufficient to analyse inter-component interactions of modular software systems.

Typically prevalent reliability models have been broadly divided into two categories [1 - 4]:

1. Black box models
2. White box models

Some researchers have reported that traditional black-box models may not be appropriate to assess the reliability of modular application constructed from a number of components [2, 3]. To address this problem, many white-box models have been proposed to analyse the components and their internal interactions [5].

The goal of white-box models is to estimate the reliability based on software structure [7]. Generally control flow graph is used to represent the Software structure constructed by some CASE tools [8]. To enable the use of white box models for reliability analysis, it is important to estimate the parameters of transition probability (edge weight) and component reliability (Node value) in the software architecture [7]. So it is considered as a directed and weighted graph. The white-box models contain two major methods, state based and path based reliability analysis. In the path based reliability analysis, all the

path from the software system will be identified and reliability of each path will be estimated by executing it in a specific number of times [8][4]. Based on all these estimated path reliabilities, final software system reliability will be predicted. But the problem here is, if the software system having more number of testing paths, that is not possible to test all the paths within a limited amount of time. That will be an exhaustive testing which is not practically possible for complex software system. To address this problem, we already have proposed one reliability analysis framework based on path testing, where the actual software system decomposed into two to three complex paths. The reliability of these complex paths was taken to predict the reliability of actual systems. But here, the path selection was done randomly which might have led to the different reliability estimation by different tester. All these issues need to be solved to estimate the accurate reliability of modular software system. The summary of above issues are, existing models failed to analyse the internal component interaction of the system, testing all the paths of the system will not be possible practically, within limited time and resources, random path selection would lead to the inaccurate estimation of the reliability of system, number of loops should be confirmed to get consistent result and the complexity of the applicability of method need to be reduced to analyse the reliability. To address these issue and also to standardize this process, we have proposed an architecture based robust reliability analysis framework of complex component based software system by considering in-degree and out-degree of each component in this paper [9]. Here, critical components / modules will be identified from the system based on in-degree and out-degree values of each component. A small subsystem will be designed using these critical components and then test path from the subsystem will be identified by applying Kal-chan path selection algorithm. This approach will be very accurate and will avoid the exhaustive path testing. Because, based on critical components core subsystem designed initially. Then this Kal-chan path selection algorithm will select the independent path from the system which covers loop testing also. The reliability of all these selected testing paths will be estimated by the standard mathematical equation. Then the minimum and maximum reliability of each software system will be estimated, followed by final system reliability will be predicted. Two real time case studies have been taken to experiment our proposed framework. The obtained results are compared with the standard existing reliability models such as CUORM [10], LCBRM [11], Chao-Jung [3] and KCW [30] which have been used as a baseline of evaluating architecture based reliability models. CUORM model is based on the modules utilization and their reliability which was used to analyse reliability. This model takes only the components used in the execution time i.e. critical components and other components are not considered. LCBRM method focused on the path based reliability estimation method, where path selection mechanism and critical node identification is not given consideration. Chao-Jung method has done the random path selection which lead to the accurate estimation of reliability. This method also have more complexity to apply in the real time application. KCW framework is based on path testing. Here the complete system has been taken as a testing unit, from which many test paths was derived, which are used to calculate the system reliability. Here the number of testing path is too high which might leads to more resource utilization and time conception. Also number of loops and branches considered for the testing here is limited which need to be improved to get accurate prediction of system reliability.

This paper has been organized in below manner. Section 2 covers the motivation of this research work. In section 3, background of this research area and existing models classification has been given. The proposed framework and its explanation has given in section 4. The section 5, covers the experiment of this framework using two case studies. The result analysis and conclusion has been given in the section 6 and 7 respectively.

2. Motivation

All This work was motivated by the deep learning of architecture based reliability modelling [12, 13] and its impact on software development process to increase the software quality [14]. Based on our studies, there are very limited reliability prediction models available for component based software systems. Most of these existing models having many assumptions for selecting the test path, computing the reliability of each path and predicting the final system reliability [15]. These reasons motivated us to design the robust framework without having any assumption to predict the reliability of component based software system. This can be applied to component based software development process and also the traditional software development process.

3. Background study

The Reliability analysis framework / modelling was proposed to analyze and predict the reliability of software systems. These frameworks can be applied in the different stages of the software development process based on its assumptions and applicability. All the present reliability models can be classified into two categories [1 - 4] i.e. Black box and White box software reliability models. The Black box reliability models focus only the functionality of the software without understanding its internal arrangement of components and behaviour [1, 3]. Based on the failure data of the system, this black box model further classified into time between failures models [16 - 19], failure count model [6, 20 - 22], fault Seeding models [23] and input domain models [24]. The White box reliability models are used to estimate the reliability by analyzing the internal structure of coding and modular interaction of the software system which can be called as architecture based reliability analysis modelling [3, 25 - 27]. These architecture based reliability modeling further classified into path based [3, 4, 28, 29], state based [10, 30], and additive models [15, 31, 32]. We have also published a reliability analysis framework based on the software architecture and path testing. Many research are going on in this field to predict the reliability of complex system. But still none of the models applicable to the all software system without assumptions [33].

4. Proposed framework

4.1 Construct the architecture of software system

This reliability prediction framework is based on the software architecture and the interconnection between each modules / components. Before applying this framework to predict the reliability of software systems, the architecture of the software system should be prepared. For the traditional software development, CASE tools can be used to get the control flow graph of the system [8]. This control flow graph can be converted into a directed graph which can represent the software architecture. For the component based software development, architecture will be designed by the software engineers based on the components to be used to build the complete system. In short, this framework will be applied to the software architecture to predict the reliability of software systems. For that, architecture is essential which can be derived from the traditional software development steps.

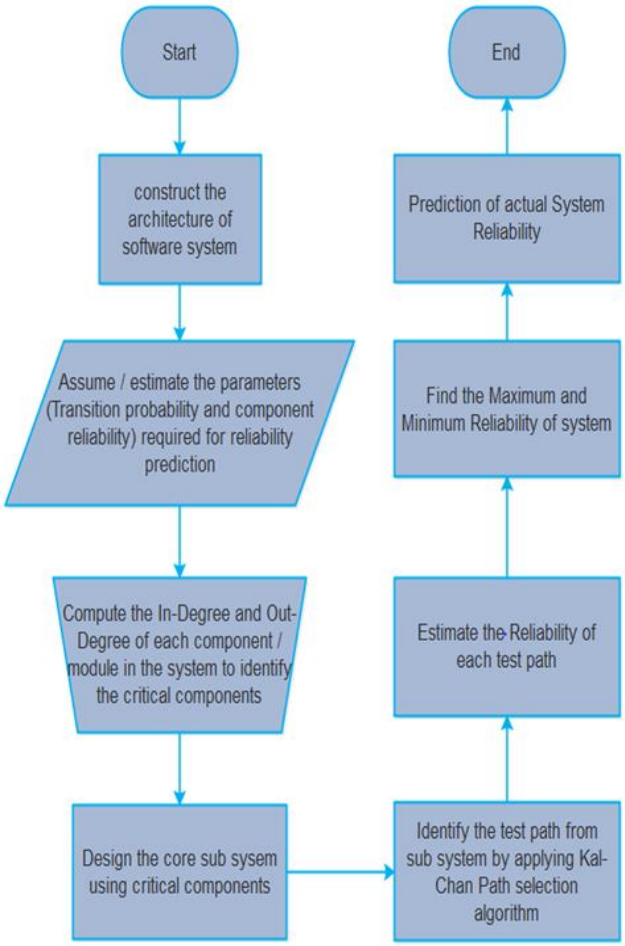

Figure. 1 Proposed reliability analysis framework

4.2 Assume / Estimate the parameters

Transition probability and component reliability are the two important parameters [3][4] which need to be available before applying this framework. Transition probability can be defined as the probability of navigating from one component to another. Fig. 1, show the steps included in the proposed reliability analysis framework of component based software system. Explanation of each step of this framework is given below. For example, in Fig. 2 the architecture of the checkout process and transition probability is given. The first node is the product or item selection component, node 2 is the checkout component, node 3 is the internet banking, node 4 is the credit/debit card payment and node five is the Case on delivery (COD) component. Here, from node 2, the user can visit any one component either 3 or 4 or 5. For example, probability of visiting the node 3, from node 2 has been given higher than the other modules, which means most of the users are visiting the node 3 after node 2.

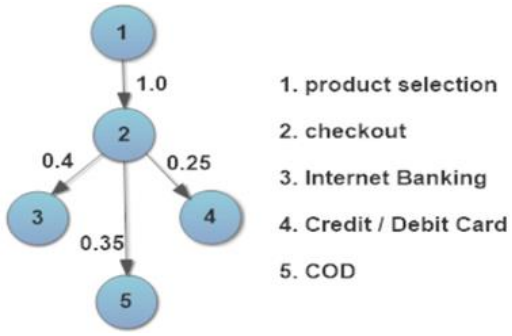

Figure. 2 Architecture of checkout process

Usually this parameter will be assumed by analysing the operational profile of the system and some analytical based methods also available in the literature for estimating these values. Component reliability is the quality of each module in the system. This can be assumed initially because not all the components are ready in the beginning of development. If the components are purchased from the vendors, then the reliability value might be available. Our proposed framework is not focusing on the estimation of these two parameters rather these two parameters are the input values of our framework. We have taken two real time case studies where transition probability and component reliabilities are already available to validate our proposed framework. These two case studies have been used by many models to validate their work generally.

4.3 Compute In-Degree and Out-Degree of each component

Once the software architecture is ready with known transition probabilities and component reliabilities values, the next core step of the framework will be applied. Here the software architecture might have n number of components and interconnection between them. Since our approach based on path testing, individual path needs to be identified for testing. But practically, we cannot test the entire system and all the paths in the system for reliability prediction. To reduce the testing overhead, core part of the system will be identified for testing, using critical components. These critical components can be identified by calculating each and every component in-degree and out-degree. These in-degree and out-degree concept which has been used in the graph theory (directed graph), to find the most linked node in the graph. Same concept applied here to identify the critical components of the system. Critical components are the one which have more connection with other components. The reliability of these critical components will affect majorly the software system reliability. Because most of the testing path might have this component as a member. Computing in-degree and out-degree of the components will be used to identify the critical nodes, so that subsystem can be designed for testing. Generally, in-degree can be defined as the number of incoming links from the other components and out-degree as the number of outgoing links from one component to the other components.

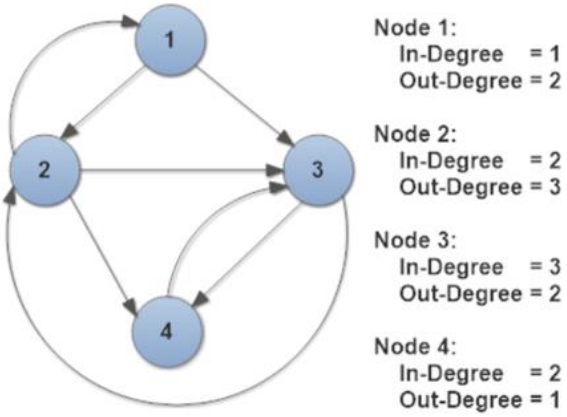

Figure. 3 In-degree and Out-degree of each node

4.4 Design of core sub-system using critical components

Subsystem can be derived from the actual software system, once the critical components are identified from the system. This sub-system will be formed by connecting all the identified critical nodes with the available links in the system. The main reason for designing this sub-system using critical components is, entire system cannot be tested within the limited time and resources. For that, the sub-system has been designed to represent the entire system with the core component which have more effects on the actual system quality. The assumption has taken and also proved, i.e. the reliability of this subsystem will be equal to the entire system reliability. This sub-system will be taken as an input for the remaining stages of our proposed framework

4.5 Identifying the test path from the sub-system using Kal-Chan Path selection algorithm

For the path selection, many researchers recommended to select the complex path from the system. So that, branch and loop structure of the system will be tested. Many results also proved that sequence path selection will have a less prediction accuracy than the branch and loop structured path. But branch and loop path have involved more complexity for estimating reliability [3, 28]. By

considering all these statement, this framework is designed by incorporating the Kal-Chan path selection algorithm [9] to derive all the test paths. The speciality of this path selection algorithm is, selecting path from the start node to the end node, which have repeated nodes to satisfy the loop testing. If the test paths are selected based on Kal-Chan algorithm, then all the loop in the system will be execute, which is also enough for setting the test adequacy criteria.

4.6 Reliability estimation of all the test path

Once all the test paths have been derived from the sub-system using Kal-Chan path selection algorithm, the reliability of each path will be estimated by applying standard, proved mathematical analysis [3].

$$Rpt_k = R_1 \times \prod_{i=2}^{n} R_i^{\prod_{j=2}^{i} P_{j-1,j}} \quad (1)$$

$Rpt_k$is the reliability of test path, k=1,2,3…
$R_i$is the reliability of component i, i=1,2,…n
n is the number of component in the sub-system
$P_{j-1,j}$ is the transition probability from node j-1 to j.

Here, all the derived test paths will be in a sequence structure including repeated node. Since it is in the sequence structure, all the components will be executed sequentially to calculate the reliability of the path. Usually, the test path derived using Kal-Chan algorithm [9] will have only one repeated node which means one loop (iteration) will be tested if it's executed. But we modified the second part of the Kal-Chan algorithm to allow more than the one loop (iteration) in the resultant test path. So that more loops and complex structure will be tested. This approach will improve the test efficiency and accuracy of the reliability prediction of the system.

4.7 Find the maximum and minimum reliability of the sub-system.

$$Min/Max.Reliability_{sub-system} = \prod_{k=1}^{m} Rpt_k \quad (2)$$

After we estimated all the test path reliabilities, maximum or minimum reliability of this sub-system can be calculated by multiplying all the estimated path reliabilities as given in Eq. (2). Multiplication of all test paths will give the maximum reliability of sub-system, if it has been designed with high quality components, otherwise this will give the minimum reliability of the system. There are two possibilities here,

1. If the multiplication of estimated test path reliabilities gives the maximum reliability, then the minimum reliability of sub-system will be calculated using Eq. (3).

$$Min.Reliability_{sub-system} = \prod_{k=1}^{n} R_{min}^{dist(k)} \quad (3)$$

2. If the multiplication of estimated test path reliabilities gives the minimum reliability, then the maximum reliability of sub-system will be calculated using Eq. (4).

$$Max.Reliability_{sub-system} = \prod_{k=1}^{n} R_{max}^{dist(k)} \quad (4)$$

n is the number of component in the software (not a sub-system)
R_{min}, R_{max} is the least and highest reliability of the component in the software system respectively
$disk(k)$, is the sum of transition probabilities from node k to R_{min} or R_{max} k=1,2,3,… n

Sometime, there is a possibility that, R_{min} or R_{max} node cannot be reached from some node n_1 . In that situation, $disk(k)$ cannot be calculated and then $n_1$should be removed from the process of calculating minimum or maximum reliability of sub-system.

4.8 Prediction of actual software system reliability

First, the sub-system reliability will be estimated based on the estimated minimum and maximum reliability. The average of minimum and maximum reliability will give the sub-system reliability as given in Eq. (5).

$$R_{sub\text{-}system} = \frac{R_{min} + R_{max}}{2} \quad (5)$$

This estimated reliability will be used to approximate the actual system reliability.

5. Experiments and explanation

Two real time case studies have been taken to validate our proposed framework. The first one is ATM bank system case study with ten components [34] and the second one is a large scale switching system which was developed at Bell laps. Fig. 4 is

the component level structure of the ATM case study. The transition probabilities and component reliabilities of this ATM structure have been assigned based on the research reported by Chao-Jung Hsu [3].

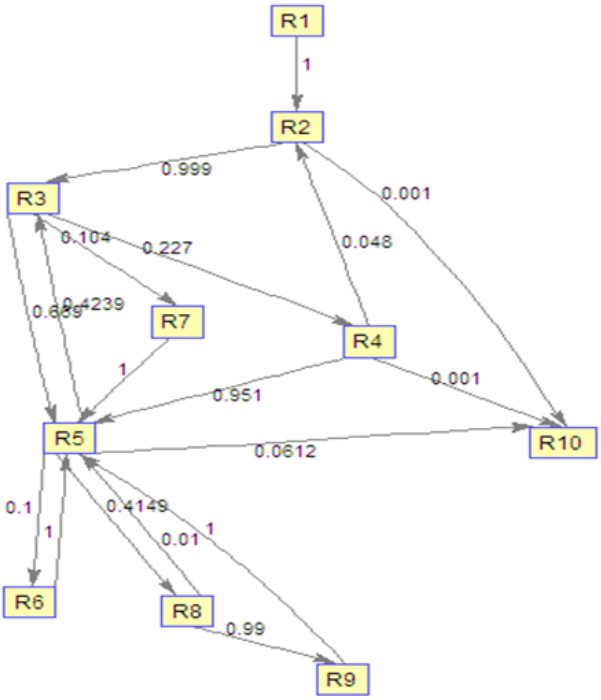

Figure. 4 ATM Case study

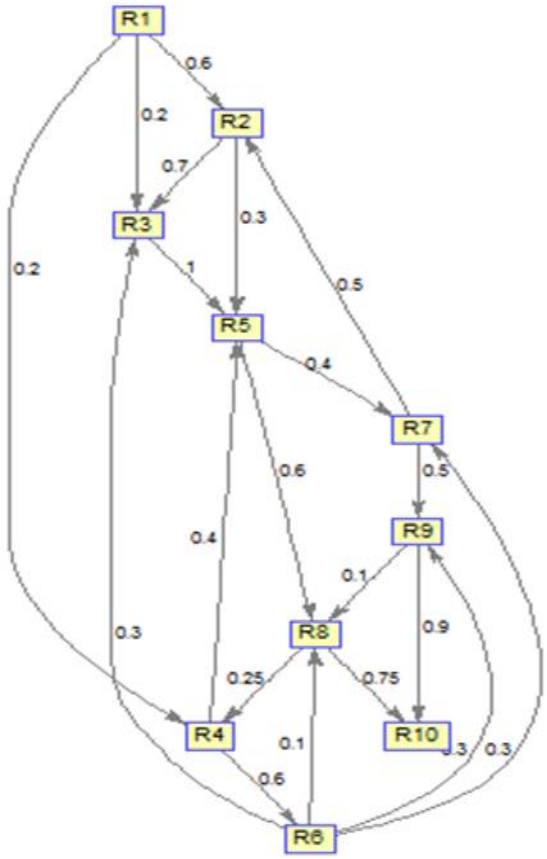

Figure. 5 Large scale switching system

In Fig. 5, large scale switching system component based structure is given [10]. This component based architecture and its parameters have been confirmed by the Gohhale [35] and Goseva [36] respectively. These two case studies are treated as a complex software system which have been acting as a baseline to validate architecture based software reliability analysis framework. Further, our proposed framework reliability prediction accuracy will be assessed by comparing the result with the standard reliability prediction frameworks such as CUORM [10], LCBRM [11], Chao-Jung Hsu [3] and KCW [4].

5.1 In-Degree, Out-Degree and critical components

In-degree and Out-degree concept has been used in the graph theory to identify the most crucial node in the directed graph. This approach has been applied here to select the critical component of software architecture.

All the components of two case studies, in-degree, out-degree and total edges is given in the Tables 1 and 2 respectively.

Table 1. ATM case study, components In-Degree and Out-Degree

Component	In-Degree	Out-Degree	Total Edges	No.of Components Connected
1	0	1	1	1
2	2	2	4	**4**
3	2	3	5	**4**
4	1	3	4	**4**
5	6	4	10	**7**
6	1	1	2	1
7	1	1	2	**2**
8	1	2	3	2
9	1	1	2	2
10	3	0	3	**3**

Table 2. Large scale switching system, components In-degree and Out-degree

Component	In-Degree	Out-Degree	Total Edges	No.of Components Connected
1	0	3	3	**3**
2	2	2	4	4
3	3	1	4	**4**
4	2	2	4	4
5	3	2	5	**5**
6	1	4	5	**5**
7	2	2	4	4
8	3	2	5	**5**
9	2	2	4	4
10	2	0	2	**2**

Based on this table value critical components are identified from two case studies. More than 50% of nodes need to be selected to form the sub-system, for that Component C2, C3, C4, C5, C7, C10, I.e., six nodes has been selected from case study 1. Here starting and ending components of a software system should be selected irrespective of considering total edges to form the sub-system.

But C1 has not been selected here as a critical node due to the following reason,

- C2 has been selected as a critical node
- C1 having only one link which is only connected to the C2

So that, according to the theory C1 and C2 can be combined as a single node.

From case study 2, components C1, C3, C5, C6, C8, C10, has been considered as a critical node to form the sub-system.

5.2 Sub-system design and test path selection

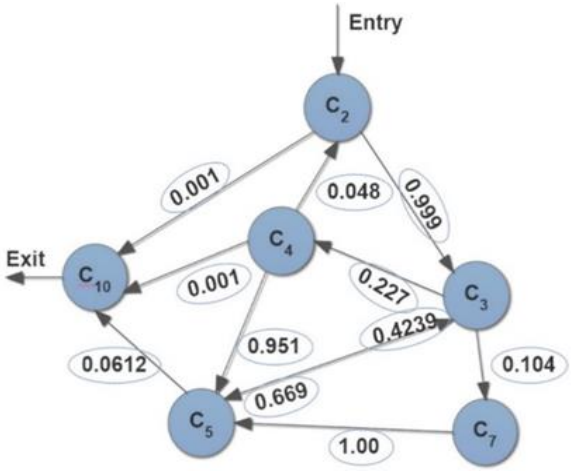

Figure. 6 Sub-system case study 1

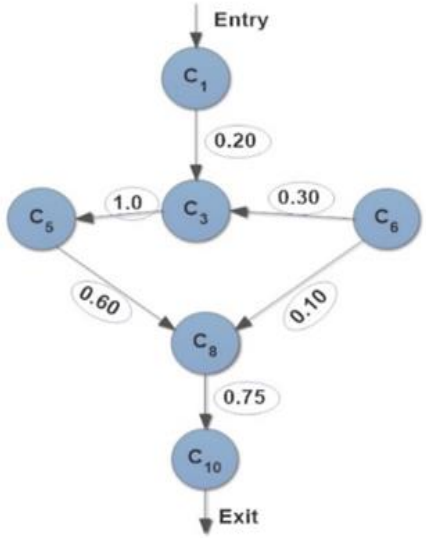

Figure. 7 Sub-system case study 2

The sub-system of ATM Bank System (Case Study 1) and Large Scale Switching System (Case Study 2) are shown in Figs. 6 and 7. These sub-system is designed with the critical components which was identified in the previous step. All these critical nodes are connected with the links available in the actual software system. As per our research, these sub-systems are representing the respective actual software system. These sub-system will be taken for the reliability estimation process. As we stated before, the reliability of these sub-system will be equal to the respective actual software system. There is no need of testing entire system to predict the reliability, testing this sub-system will be adequate to predict the actual software system reliability. This approach will be used to reduce the testing overhead, time and resources required for testing. This approach will also give the accurate prediction of reliability compared to the other standard baseline models [3, 4, 10, 11].

Next, test path will be identified using Kal-Chan path selection algorithm [9]. According to the Kal-Chan path selection algorithm, there will be two phases involved in the path selection process. These two phases will be a, path selection from source node to the intermediate node and intermediate node to the destination node. So, before starting this process, intermediate node should be identified from both the sub-system. Here, we have considered C5 and C8 are an intermediate node of the sub-systems of ATM Bank System and Large Scale Switching System respectively. As per Kal-Chan algorithm, C5 and C8 have been considered as an intermediate node, since it has the more connection with other components. In the first phase of Kal-Chan algorithm, all the nodes in the sub-system will be acting as a source node, except intermediate and last node of sub-system. The path between source nodes to intermediate node will be derived in the first phase of the algorithm. In the second phase, path between an intermediate node to the destination node will be derived. Usually in the path selection process, visited node will not be visited again. But here, we have modified the second phase of the algorithm alone to allow the visited node to be visited again. Anyhow, this modification will be allowed for only one node. This changes have been made in the algorithm to test more than one loop (iteration) of the system for the effective testing. Finally, these two paths will be combined to form the test path.

In the ATM Case study, C2, C3, C4, C7 will be a source node and the path derived from source node to the intermediate is given in Fig. 8, similarly Intermediate node to last node is given in Fig. 9.

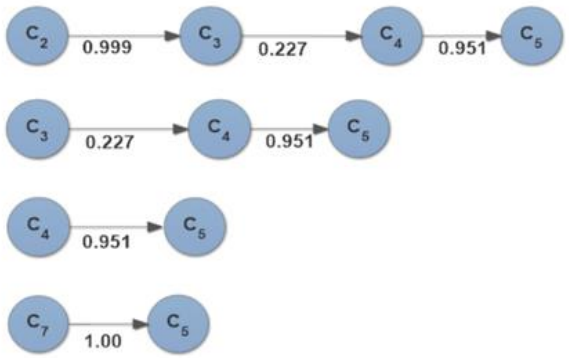

Figure. 8 ATM subsystem. Source to intermediate path

Figure. 9 ATM subsystem, Intermediate to last node path

Table 3. Test Path of ATM Sub-system

Test path	Path Structure
Path 1 (Tp1)	C2->C3->C4->C5->C3->C5->C10
Path 2 (Tp2)	C3->C4->C5->C3->C5->C10
Path 3 (Tp3)	C4->C5->C3->C5->C10
Path 4 (Tp4)	C7-> C5->C3->C5->C10

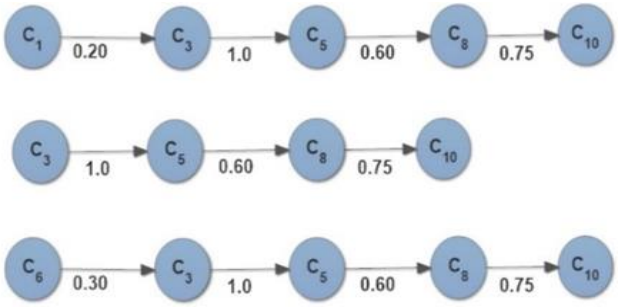

Figure. 10 Test path of case study 2 - sub-system

The final testing path of ATM sub-system after combining these two structures is given in Table 3.

Similarly, for case study 2 (Large scale switching system) subsystem, final derived test path is given in Fig. 10.

5.3 Test paths reliability estimation and system reliability prediction

Once the test paths have been derived from two case studies, the reliability of each test path will be estimated using the Eq. (1). Then the maximum and minimum reliability of each sub-system will be estimated using Eqs. (2) and (3). These reliabilities will be used to calculate the sub-system reliability by averaging the minimum and maximum reliability as shown in Eq. (5), followed by the actual software system reliability will be approximated based on the results obtained.

Table 4. Case study 1 sub-system. Test path reliability, maximum & minimum reliability and final reliability

Test path	Estimated Path Reliability	Max. Reliability	Min. Reliability	Final Reliability
Tp1	0.905	0.7092	0.2437	0.4764
Tp2	0.932			
Tp3	0.912			
Tp4	0.922			

Table 5. Case study 2 sub-system. Test path reliability, maximum & minimum reliability and final reliability

Test path	Estimated Path Reliability	Max. Reliability	Min. Reliability	Final Reliability
Tp1	0.992	0.9075	0.7778	0.8426
Tp2	0.951			
Tp3	0.962			

The estimated path reliability of two case studies are given in the Tables 4 and 5.

This final reliability will be taken as a complete system reliability. Further the accuracy of our proposed framework result will be assessed by comparing with the standard reliability models CUORM, LCBRM, Chao-Jung and KCW which have been used by many researchers in this field. The discussion and comparison of our result are given in the below section.

6 Result and discussion

Our proposed framework based on path testing experimented by two case studies [35, 37] and the result has been obtained. This result shows that, our proposed framework has a high correlation to the standard existing models and the actual software reliability. The comparison of proposed framework to the CUORM, LCBRM, Chao-Jung and KCW framework have been given in Table 6 and Fig. 11.

Table 6. A proposed framework estimated reliability comparison with other models

Reliability Models	Reliability Value Case Study 1	Reliability Value Case Study 2
Proposed Model	0.4764	0.8426
CUORM Reliability	0.556	0.829
LCBRM Reliability	0.448	0.827
Chao-Jung Reliability	0.428	0.846
KCW Framework	0.458	0.828
Actual Software Reliability	0.441	0.826

Table 7. Accuracy of proposed framework compared with other models using RE

Real Time Example	RE with Actual Reliability	RE with CUORM	RE with LCBRM	RE with Chao-Jung	RE with KCW
Case Study 1	0.0802	0.1431	0.0633	0.1130	0.0401
Case Study 2	0.0200	0.0164	0.0188	0.0040	0.0176
Mean of RE (Case Study 1) 0.0879			Mean of RE (Case Study 2) 0.0153		

Table 8. Relative error and mean RE of all the existing model

	Relative Error of CUORM					
	RE with Actual Reliability	**RE with Proposed Framework**	**RE with LCBRM**	**RE with Chao-Jung**	**RE with KCW Framework**	**Mean of RE**
Case Study 1	0.2607	0.1670	0.2410	0.2990	0.2139	0.2363
Case Study 2	0.0036	0.0161	0.0024	0.0200	0.0012	0.0086
	Relative Error of LCBRM					
	RE with Actual Reliability	**RE with Proposed Framework**	**RE with CUORM**	**RE with Chao-Jung**	**RE with KCW Framework**	**Mean of RE**
Case Study 1	0.0158	0.0596	0.1942	0.0467	0.0218	0.0676
Case Study 2	0.0012	0.0185	0.0024	0.0224	0.0012	0.0091
	Relative Error of Chao-Jung					
	RE with Actual Reliability	**RE with Proposed Framework**	**RE with CUORM**	**RE with LCBRM**	**RE with KCW Framework**	**Mean of RE**
Case Study 1	0.0294	0.1015	0.2302	0.0446	0.0655	0.0942
Case Study 2	0.0242	0.0040	0.0205	0.0229	0.0217	0.0186
	Relative Error of KCW Framework					
	RE with Actual Reliability	**RE with Proposed Framework**	**RE with CUORM**	**RE with LCBRM**	**RE with Chao-Jung**	**Mean of RE**
Case Study 1	0.0385	0.0386	0.1762	0.0223	0.0700	0.0691
Case Study 2	0.0024	0.0173	0.0012	0.0012	0.0212	0.0086

To compare the accuracy of our proposed framework with the actual software reliability and the other existing models, the relative error [37, 38] can be used which is defined as,

$$RE = \frac{R_p - R_a}{R_a} \quad or \quad \frac{R_p}{R_a} - 1 \qquad (6)$$

R_a is the actual software reliability / existing model reliability for comparison.

R_p is the estimated reliability of the proposed framework.

Relative Error of the proposed framework, compared with actual software reliability and other existing model which is given in Table 7 and Fig. 12.

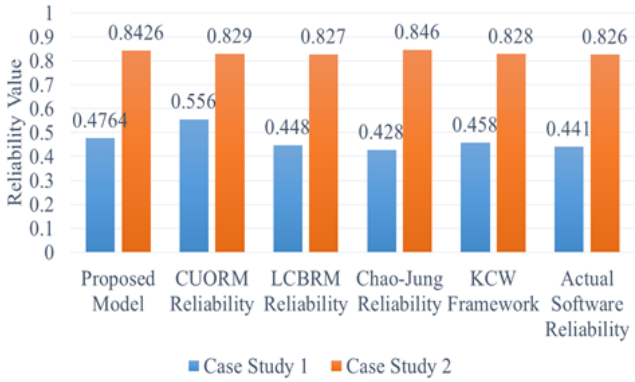

Figure. 11 Proposed framework reliability of case study 1 and 2, compared with existing models

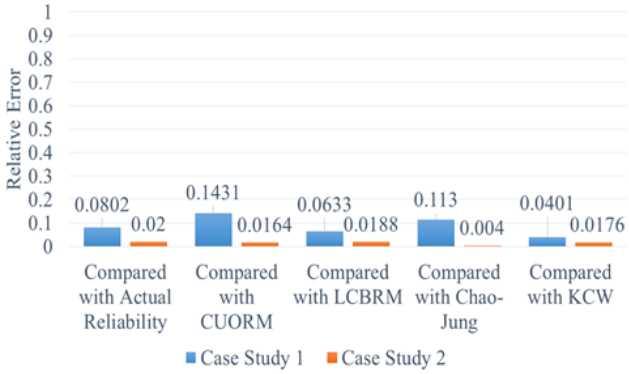

Figure. 12 Proposed framework RE comparison with other models (Case Study 1 & 2)

Table 9. Comparison of mean of relative error (RE)

Reliability Models	**Mean of RE (Case Study 1)**	**Mean of RE (Case Study 2)**
Proposed Model	0.0879	0.0153
CUORM Reliability	0.2363	0.0086
LCBRM Reliability	0.0676	0.0091
Chao-Jung Reliability	0.0942	0.0186
KCW Framework	0.0691	0.0086

Similarly, Relative Error (RE) and the mean of Relative Error has been calculated for the existing models and the result is shown in Table 8. Then, the mean of all the models Relative Error is compared in Table 9 against the proposed framework. The result shows that, our proposed framework having more and acceptable accuracy compared to the other standard models, and can also be applied to predict the reliability of software system effectively.

7 Conclusion

General issues in the prevalent architecture based software reliability analysis methods are less test coverage, random path selection, complexity in solving mathematical equation, and more number of testing path selection and its applicability in the real software system. All these issues are addressed by the proposed framework.

Our proposed reliability analysis framework is based on graph theory, which can be used extensively in the critical and complex component based software system to predict the reliability in the early phase of software development process.

This framework used the in-degree and out-out-degree of each module or component to reduce the size of software system for testing. The number of paths for testing have been reduced due to this sub-system design which would help the tester to reduce the effort and time to be spent for testing phase. Kal-chan path selection algorithm was used to identify the test path from the sub-system. The derived paths using this algorithm were effective, since it has the repeated node in the path which was helpful to test the loop structure of sub-system. This model experimented with two case studies and the accuracy of our framework compared with standard models which have been used as a baseline for evaluating models. The relative error of each model and its mean value of relative error is calculated to prove the accuracy of our model. The result shows that, our proposed method giving acceptable accuracy compared to the other models and this can be applied to real time software system. Solving the mathematical equation of proposed method is also simple. During the sub-system design, there may be the possibility to design the two more sub-systems, if there is a more critical node in the system. That part needs more attention since it affects the system reliability prediction accuracy. The study of all the critical nodes and the sub-system design will be done and will be checked with different case studies in the future.

References

[1] K. Cai, C. Wen, and M. Zhang, "A critical review on software reliability modeling", *Reliability Engineering & System Safety*, Vol. 32, No. 3, pp. 357-371, 1991.

[2] S. Gokhale, "Architecture-Based Software Reliability Analysis: Overview and Limitations", *IEEE Transactions on Dependable and Secure Computing*, Vol. 4, No. 1, pp. 32-40, 2007.

[3] C. Hsu and C. Huang, "An Adaptive Reliability Analysis Using Path Testing for Complex Component-Based Software Systems", *IEEE Transactions on Reliability*, Vol. 60, No. 1, pp. 158-170, 2011.

[4] S. Kaliraj, N. Chandru, and W. Amitabh, "A Reliability Analysis Framework of Component Based Software System Using Kal-Chan Path Selection Algorithm", *International Review on Computers and Software*, Vol. 8, No. 2, pp. 605-612, 2013.

[5] S. Yuanjie, X. Yang, X. Wang, C. Huang, and A. Kavs, "An architecture-based reliability estimation framework through component composition mechanisms", In: *Proc. of the 2nd International Conference on Computer Engineering and Technology*, Vol. 2, pp. V2-165, 2010.

[6] Shooman and L. Martin, "Software reliability-Measurement and models", In: *Proc. of the*

Annual International Symposium on Reliability and Maintainability, pp. 485-491, 1975.

[7] D. Wang, H. Ning, and Y. Ming, "Reliability analysis of component-based software based on relationships of components", In: *Proc. of the IEEE International Conference on Web Services*, pp. 814-815, 2008.

[8] H. Chao-Jung and C. Huang, "Integrating path testing with software reliability estimation using control flow graph", In: *Proc. of the 4th IEEE International Conference on Management of Innovation and Technology*, pp. 1234-1239, 2008.

[9] S. Kaliraj and N. Chandru, "Introducing path selection algorithm in reliability analysis of component based software system", In: *Proc. of the 6th International Conference on Quality, Reliability, Infocom Technology and Industrial technology management*, Vol. 6, 2012.

[10] R. Cheung, "A User-Oriented Software Reliability Model", *IEEE Transactions on Software Engineering*, Vol. 6, No. 2, pp. 118-125, 1980.

[11] J. Lo, C. Huang, I. Chen, S. Kuo, and M. Lyu, "Reliability assessment and sensitivity analysis of software reliability growth modeling based on software module structure", *Journal of Systems and Software*, Vol. 76, No. 1, pp. 3-13, 2005.

[12] W. Wang, D. Pan, and M. Chen, "Architecture-based software reliability modeling", *Journal of Systems and Software*, Vol. 79, No. 1, pp. 132-146, 2006.

[13] C. Zhang, Y. Ma, X. Wang and R. Wang, "Software Architecture Modelin,g and Reliability Evaluation Based on Petri Net," In: *Proc. of the 2017 International Conference on Dependable Systems and Their Applications*, pp. 51-56, 2017.

[14] S. Kaliraj, N. Premkumar, and A. Bharathi, "The Novel Life Cycle Model for Component Based Software System Based on Architecture Quality Using KCW Framework", *International Journal of Information Technology and Computer Science*, Vol. 6, No. 9, pp. 74-79, 2014.

[15] K. Goseva-Popstojanova and K. Trivedi, "Architecture based software reliability", In: *Proc. of the International Conference on Applied Stochastic System Modeling*, 2000.

[16] Z. Jelinski and P. Moranda, "Software reliability research", *Statistical computer performance evaluation*, pp. 465-484, 1972.

[17] G. Schick and R. Wolverton, "Assessment of software reliability", In: *Proc. of Vorträge der jahrestagung 1972 dgor/papers of the annual meeting*, pp. 395-422, 1973.

[18] P. Moranda, "Prediction of software reliability during debugging", In: *Proc. of the Annual Reliability Maintenance Symposium*, 1975.

[19] B. Littlewood and J. Verrall, "A Bayesian Reliability Growth Model for Computer Software", *Applied Statistics*, Vol. 22, No. 3, pp. 332-346, 1973.

[20] A. Goel and K. Okumoto, "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures", *IEEE Transactions on Reliability*, Vol. 28, No. 3, pp. 206-211, 1979.

[21] J. Musa, "A theory of software reliability and its application", *IEEE Transactions on Software Engineering*, Vol. 1, No. 3, pp. 312-327, 1975.

[22] J. Musa and K. Okumoto, "A logarithmic Poisson execution time model for software reliability measurement", *In: Proc. of the 7th international conference on Software engineering*, pp. 230-238, 1984.

[23] H. Mills, "On the statistical validation of computer programs", *IBM Federal Systems Division Report*, 1972.

[24] C. Ramamoorthy and F. Bastani, "Software Reliability—Status and Perspectives", *IEEE Transactions on Software Engineering*, Vol. 8, No. 4, pp. 354-371, 1982.

[25] G. Katerina and M. Hamill, "Architecture-based software reliability: Why only a few parameters matter?", In: *Proc. of the 31st Annual International Computer Software and Applications Conference*, Vol. 1, pp. 423-430, 2007.

[26] S. Gokhale and K. Trivedi, "Reliability prediction and sensitivity analysis based on software architecture", In: *Proc. of the 13th International Symposium on Software Reliability Engineering*, pp. 64-75, 2003.

[27] K. Goševa-Popstojanova and K. Trivedi, "Architecture-based approach to reliability assessment of software systems", *Performance Evaluation*, Vol. 45, No. 2-3, pp. 179-204, 2001.

[28] S. Krishnamurthy and A. Mathur, "On the estimation of reliability of a software system using reliabilities of its components", In: *Proc. of the Eighth International Symposium on Software Reliability Engineering*, pp. 146-155, 1997.

[29] M. Shooman, "Structural models for software reliability prediction", In: *Proc. of the 2nd international conference on Software engineering*, pp. 268-280, 1976.

[30] S. Gokhale, W. Wong, K. Trivedi, and J. Horgan, "An analytical approach to architecture-based software reliability prediction", In: *Proc. of the IEEE International Symposium on Computer Performance and Dependability*, pp. 13-22, 1998.
[31] W. Everett, "Software component reliability analysis", In: *Proc. of the IEEE Symposium on Application-Specific Systems and Software Engineering and Technology*, pp. 204-211, 1999.
[32] M. Xie and C. Wohlin, "An additive reliability model for the analysis of modular software failure data", In: *Proc. of the Sixth International Symposium on Software Reliability Engineering*, pp. 188-194, 1995
[33] M. Lyu, Handbook of software reliability engineering. New York: McGraw-Hill, 1996.
[34] W. Wang, Y. Wu, and M. Chen, "An architecture-based software reliability model", In: *Proc. of the International Symposium on Dependable Computing*, pp. 143-150, 1999.
[35] S. Gokhale and M. R.-T. Lyu, "A simulation approach to structure-based software reliability analysis", *IEEE Transactions on Software Engineering*, Vol. 31, No. 8, pp. 643-656, 2005.
[36] K. Goseva-Popstojanova and S. Kamavaram, "Assessing uncertainty in reliability of component-based software systems", In: *Proc. of the 14th International Symposium on Software Reliability Engineering*, pp. 307-320, 2003
[37] J. Musa, Software reliability: measurement, prediction, application. New York: McGraw-Hill Book Co. song gang, 1987.
[38] C. Huang and C. Lin, "Software Reliability Analysis by Considering Fault Dependency and Debugging Time Lag", *IEEE Transactions on Reliability*, Vol. 55, No. 3, pp. 436-450, 2006.