International Journal of Mathematical, Engineering and Management Sciences
Vol. 5, No. 2, 309-318, 2020
https://doi.org/10.33889/IJMEMS.2020.5.2.025

A Non-Parametric Approach for Survival Analysis of Component-Based Software

Sandeep Chopra
Department of Computer Application & Information Technology,
S.G.R.R. University, Dehradun, Uttrakhand, India.

Lata Nautiyal
Department of Computer Science,
University of Bristol, Bristol, United Kingdom.
Corresponding author: lata.nautiyal@bristol.ac.uk

Preeti Malik
Department of Computer Science and Engineering,
Graphic Era Deemed to be University, Dehradun, Uttrakhand, India.

Mangey Ram
Department of Mathematics; Computer Science and Engineering,
Graphic Era Deemed to be University, Dehradun, Uttrakhand, India.

Mahesh K. Sharma
Department of Computer Application & Information Technology,
Amrapali Institute, Haldwani, Uttrakhand, India.

(Received September 11, 2018; Accepted August 27, 2019)

Abstract
Reliability of a software or system is the probability of system to perform its functions adequately for the stated time period under specific environment conditions. In case of component-based software development reliability estimation is a crucial factor. Existing reliability estimation model falls into two broad categories parametric and non-parametric models. Parametric models approximate the model parameters based on the assumptions of fundamental distributions. Non-parametric models enable parameter estimation of the software reliability growth models without any assumptions. We have proposed a novel non-parametric approach for survival analysis of components. Failure data is collected based on which we have calculated failure rate and reliability of the software. Failure rate increases with the time whereas reliability decreases with the time.

Keywords- Component-based software, Failure, Survival analysis, Non-parametric method, Reliability.

1. Introduction
Now a days, software development organization of industry has become progressively dependent on third party for functionality. This is due to financial and time-to-market consideration. These third party's software or components are then integrated to form complete software as per the needs of the customer. Components are high-quality and pretested software entities. This methodology of software development is called Component-based software engineering (CBSE) (Gayen and Misra, 2008).

CBSE plays an important role in this era of software. CBSE comprises of application and component engineering. One of the grimmest problems for successful CBSE is its reliability estimation. Analyzing the reliability of software is crucial for predicting software field failure (Tyagi and Sharma, 2012). The term reliability can be defined as "Probability of a system to perform its functions correctly for a specified period of time."

Reliability is measured with respect to time. Traditional methods for estimating reliability can't be applied to component-based software (CBS) applications. There are various methods already proposed by researchers. These approaches for reliability estimation involve two steps (Goseva-Popstojanova and Trivedi, 2003): Approximating the reliability of distinct components, and the reliability of system. Nautiyal and Preeti (2016) have proposed an evaluation process for certification of component based software. Certification is performed at component as well as system level. Author has used unstructured weighting technique to certify the system or component. The author Gokhale (2007) has proposed an overview of the existing research in the area of architecture-based software reliability analysis and critically examined the growing size and complexity of software applications.

Reliability estimation models falls into three categories: state-based, path-based and additive models (Singh et al., 2001; Yacoub and Ammar, 2002). To estimate reliability, State-based models observe the flow of control among components. The models assume that components may be faulty autonomously and current behavior of a component doesn't depend on its earlier behavior. Failure is modeled as Non-Homogeneous Poisson Process (NHPP). The limitation of these models is that the component's failure probability cannot be constant because failure rate may be high for frequently used components. So, the assumption of constant failure rate cannot lodge this fact.

Path-based models take into account the possible execution paths for estimating the system reliability. Experiments and algorithms are two ways to obtain different paths. Path's reliability is defined in terms of a function of the reliabilities of the components along that particular path. Reliability of the system is the average of reliabilities of all paths. Third category of models is additive model. Failure data of the component is used to estimate the system reliability. Additive models study growth of software reliability. Additive models do not explicitly take into account architecture of the software. Reliability of a system can be estimated from failure rate by using many techniques. We can categorize these techniques into two broad categories:

- Non-parametric techniques
- Parametric techniques.

Non-Parametric methods are commonly used for estimating the reliability characteristics. These methods are simple to use. The constraint is that the results cannot be precisely generalized outside the last reported failure rate. In Parametric techniques, the failure rate is to fit to a statistical distribution (exponential, normal, Weibull, or lognormal). The resultant model can be used for efficient calculation of reliability parameters for the entire lifetime of the system.

We have proposed a non-parametric additive model to estimate the reliability of the CBS. In proposed approach the reliability estimation is based on failure data of the components. Failure data of a CBS is collected and accordingly reliability is computed. Probability of failure is used to represent the failure behavior. Remaining paper is organized as follows; next section discusses the related work done in this area. Section 3 consists of proposed approach. Final section includes conclusion of the paper.

2. Related Work
Software reliability model falls into two main categories: parametric and non-parametric models (Lakshmanana and Ramasamy, 2015). Parametric models approximate the model parameters based on the assumptions of fundamental distributions. These models can be further divided into three types: NHPP, Markovian models (Whittaker et al., 2000) and Bayesian models. Non-parametric models enable parameter estimation of the software reliability growth models (SRGM) without any assumptions. Non-parametric methods yield models with better analytical accurateness than parametric models (Karunanithi et al., 1992).

The author, Su et al. (2007) have proposed a fuzzy-logic based model to estimate the reliability of CBS. Author considers four factors that affect the reliability, reusability and operational profile in case of component reliability and component dependency and application complexity to estimate interface reliability. Zhang et al. (2009) have introduced the concept of reliability estimation using architecture-based model. This approach for reliability evaluation can be applied in design phase. This approach assumes that the overall reliability is related to the individual component's reliability.

The author Isaac (1995) focused on the main two points i.e. risk assessment and risk control where risk assessment helps a manager to make judgment about his future and helps others to overcome their errors. This paper also highlighted on ten points that should be kept in mind while using risk management techniques. Bowers and Khorakian, (2014) has proposed new method which is quite similar to other projects which include failure rate and emphasizes on creativity. Without risk management it is difficult to achieve success. But an excessive risk can also hamper the creativity. So, to be on the safer side one should use risk management technique.

The authors, Wang and Huang (2008) have offered reliability analysis based on rewrite logic technique. This method is based on analysis of operation profile and specifications. Rewrite language Maude is used to execute these specifications. Execution process is used to calculate transition probabilities and statistically analyze the expected numbers of components, which will be visited. Critical components can also identified by this algorithm.

Weiss and Weyuker (1988) have provided the approach in faces a problem of test case selection from a specific input domain since there were no strategies concerning selection of test cases and occurrence of operational errors. Gayen and Misra (2009) have solved this problem by dividing the input field into operational error subfield and logical subfield. Path coverage based testing methodology is used to select test cases and to predict the reliability in the logical sub-domain. To obtain the actual input domain based reliability this value is multiplied with the probability of non-occurrence in the operational error sub-domain.

Yacoub et al. (2004) have proposed Scenario-based reliability evaluation method. This approach presents component dependency graphs that can be extended for complex distributed systems. The approach is constructed on scenarios which can be seized with sequence diagrams. It means that this approach can be automated. A disadvantage of this approach is that it does not take into account the failure dependencies among the components.

Gokhale et al. (1998) have discussed an approach in which author assumes that the application can be represented as a control flow graph. Component failures are randomly generated for simulation. A programmatic procedure is used to return the inter failure arrival time for a particular component. Simulation failures use these failure and repair rate while executing the application and its reliability is estimated. Component interface and link failure are not considered while simulation is being performed.

Lo (2010) has proposed a software reliability estimation model based on a Support Vector Machine (SVM) and Genetic Algorithm (GA). Advantage of this model is that it does not depend on failure data much. This approach states that topical failure data itself is enough for estimating reliability. Reliability estimation parameters for the SVM are determined by the GA. Goswami and Acharya (2009) have considered component usage ratio (CUR) for reliability analysis of CBS. Mathematical formulas are used to compute CUR. Due to the suppleness of the CUR, this technique may be used in real-time applications. Everett (1999) proposes a six step process for software reliability; dividing software into components, Characterize the component, define usage of components, Model the reliability of discrete components, Superimpose the reliability of components, Component analysis through testing.

3. Proposed Approach

Let t_1, t_2, t_3…represent the time of failure of component. Also let n_1, n_2, n_3… symbolize the number of component failure that happen at each of these times, and let r_1, r_2, r_3… be the corresponding number of components lasting. It means $r_2 = r_1 - n_1$, $r_3 = r_2 - n_2$, etc. We know that the probability of lasting beyond time t_2 i.e. $(P(T>t_2))$ depends on probability of lasting beyond time t1 i.e. $(P(T>t_1))$. Similarly, probability of lasting beyond time t_3 depends on probability of lasting beyond time $t_2$etc. We can use this recursive relation to iteratively build a numerical estimate R (t) of the true survival function R(t).

For any time $t \in [0, t_1)$, we have R (t) = P (T >t) = "Probability of surviving beyond time t" = 1, because no failures have occurred as yet. Therefore, for all t in this interval, let R (t)=1.

Note: For any two events A and B, **P (A and B) = P(A) × P(B | A)**.
Let A = "survive to time t_1" and B = "survive from time t_1 to beyond some time t before t_2". As both events occurs therefore equivalent time of the event "A and B" = "survive beyond time t before t_2," i.e, "T >t." Hence, the following condition holds.

For any time $t \in [t_1, t_2)$, we have…

$$R(t) = P(T > t) = P(\text{survive in } [0, t_1)) \times P(\text{survive in } [t_1, t] \mid \text{survive in } [0, t_1)),$$

$$R(t) = 1 \times \left(\frac{r_1 - n_1}{r_1}\right)$$

$$S(t) = \left(1 - \frac{n_1}{r_1}\right).$$

For any time $t \in [t_2, t_3)$, we have…

$S(t) = P(T > t) = P(\text{survive in } [t_1, t_2)) \times P(\text{survive in } [t_2, t] \mid \text{survive in } [t_1, t_2))$,

$$= (1-\frac{n_1}{r_1}) \text{ x } (\frac{r_2 - n_2}{r_2})$$

$$S(t) = (1-\frac{n_1}{r_1}) \text{ x } (1-\frac{n_2}{r_2})$$

In general, for $t \in [t_j, t_{j+1})$, $j = 1, 2, 3…$ we have…

$$S(t) = (1-\frac{n_1}{r_1})\,(1-\frac{n_2}{r_2}) \ldots\ldots\ldots\ldots (1-\frac{n_j}{r_j}) = \prod_{i=1}^{j}(1-\frac{n_i}{r_i}) \text{ where,}$$

r_j = the number of component failures in the interval j,
n = the total number of components,
t_j = time taken for d_j failure,
n_j = the operating components in the interval j i.e. $n - \Sigma r_j$.

3.1 Steps for Survival Analysis of CBS

Proposed approach comprises of four phases. Figure 1 shows the diagram of proposed approach. Four phases are as follows:

(i) *Take a CBS and Test it:* We have coded a CBS comprises of 30 components. These components don't perform any function but only prints something on the screen. We consider a component is failed if at some time it is not printing its statement on the screen. Each component runs as a thread of java program. For introducing failure we have stopped the particular thread.

(ii) *Collect Time-To-Failure and Number of Components Failed:* Table 1 shows the failure data collected in testing this CBS.

(iii) *Calculate Failure Rate:* Third column in Table 2 gives calculated failure rate. Failure rate vs. time graph (in Figure 2) shows failure rate increases as the time increases.

(iv) *Calculate Reliability:* Last column of Table 2 in gives reliability calculated by using the proposed approach. Reliability Vs. time graph (in Figure 3) shows reliability decrease as the time increases.

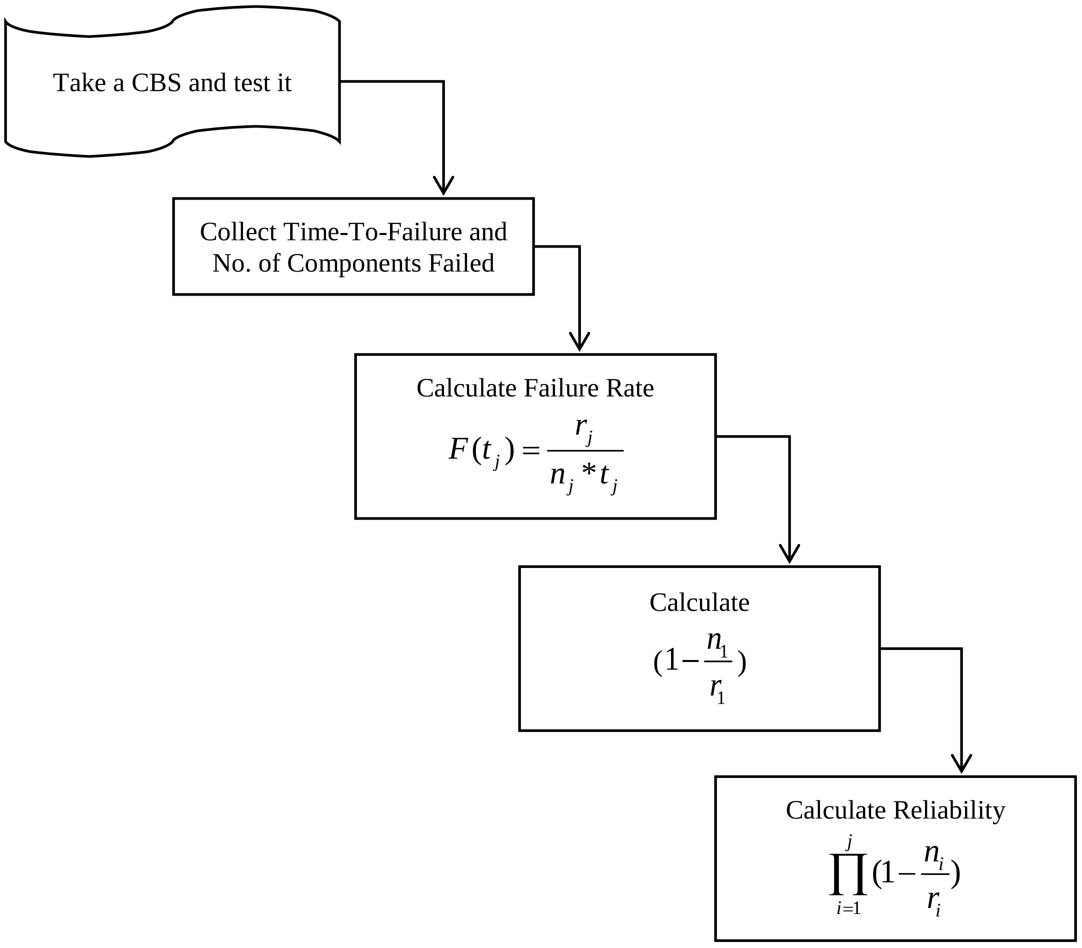
Take a CBS and test it
Collect Time-To-Failure and No. of Components Failed
Calculate Failure Rate
$F(t_j) = \frac{r_j}{n_j * t_j}$
Calculate
$(1 - \frac{n_1}{r_1})$
Calculate Reliability
$\prod_{i=1}^{j} (1 - \frac{n_i}{r_i})$

Figure 1. Flow chart for proposed approach

3.2 Reliability Analysis

Reliability of the software is the ability of the software to perform the required function, under some scenario or pre-defined condition for a stated period of time. It is usually defined as the probability of failure free operation for a specified time, in specified environment for a specific purpose. It is the important attribute of software quality. Reliability is basically categorized into two parts

- Hardware Reliability
- Software Reliability

Hardware reliability means, what is the probability of hardware component failing and how long does it take to repair that component? Software reliability is the probability that the software system will function properly without failure over a certain period of time. This section presents reliability analysis a system with 31 components.

Table 1. Time-to-failure of CBS

Rank	Time to Failure	No. of Component Failure (r_j)
1	0	0
2	80	1
3	81	1
4	83	1
5	84	1
6	87	1
7	90	1
8	91	1
9	93	1
10	94	1
11	95	1
12	97	1
13	99	1
14	100	1
15	101	1
16	102	1
17	103	1
18	107	1
19	110	1
20	111	1
21	113	1
22	117	1
23	120	1
24	122	1
25	123	1
26	125	1
27	127	1
28	129	1
29	132	1
30	134	1
31	137	1

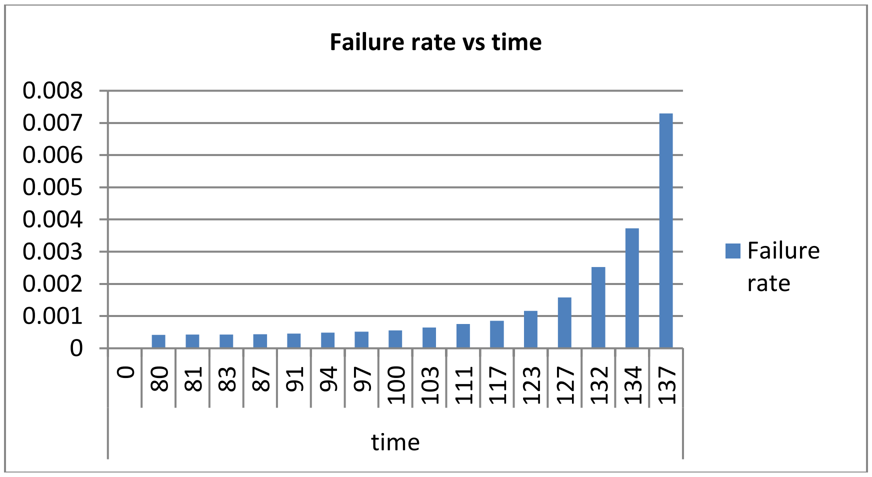

Figure 2. Failure rate vs. time graph

Table 2. Calculated values of failure rate and reliability using proposed approach

Rank	Time To Failure	No. of Component Failure (r_j)	No. of components at the beginnings of the time (n_j)	Failure Rate F(t_j)	$(n_j - r_j) / n_j$	Reliability $\{\prod (n_j - r_j) / n_j \}$
1	0	0	30	0	1	1.000
2	80	1	30	0.000417	0.966667	0.967
3	81	1	29	0.000426	0.965517	0.933
4	83	1	28	0.000430	0.964286	0.900
5	84	1	27	0.000441	0.962963	0.867
6	87	1	26	0.000442	0.961538	0.833
7	90	1	25	0.000444	0.960000	0.800
8	91	1	24	0.000458	0.958333	0.767
9	93	1	23	0.000468	0.956522	0.733
10	94	1	22	0.000484	0.954545	0.700
11	95	1	21	0.000501	0.952381	0.667
12	97	1	20	0.000515	0.950000	0.633
13	99	1	19	0.000532	0.947368	0.600
14	100	1	18	0.000556	0.944444	0.567
15	101	1	17	0.000582	0.941176	0.533
16	102	1	16	0.000613	0.937500	0.500
17	103	1	15	0.000647	0.933333	0.467
18	107	1	14	0.000668	0.928571	0.433
19	110	1	13	0.000699	0.923077	0.400
20	111	1	12	0.000751	0.916667	0.367
21	113	1	11	0.000805	0.909091	0.333
22	117	1	10	0.000855	0.900000	0.300
23	120	1	9	0.000926	0.888889	0.267
24	122	1	8	0.001025	0.875000	0.233
25	123	1	7	0.001161	0.857143	0.200
26	125	1	6	0.001333	0.833333	0.167
27	127	1	5	0.001575	0.800000	0.133
28	129	1	4	0.001938	0.750000	0.100
29	132	1	3	0.002525	0.666667	0.067
30	134	1	2	0.003731	0.500000	0.033
31	137	1	1	0.007299	0	0.000

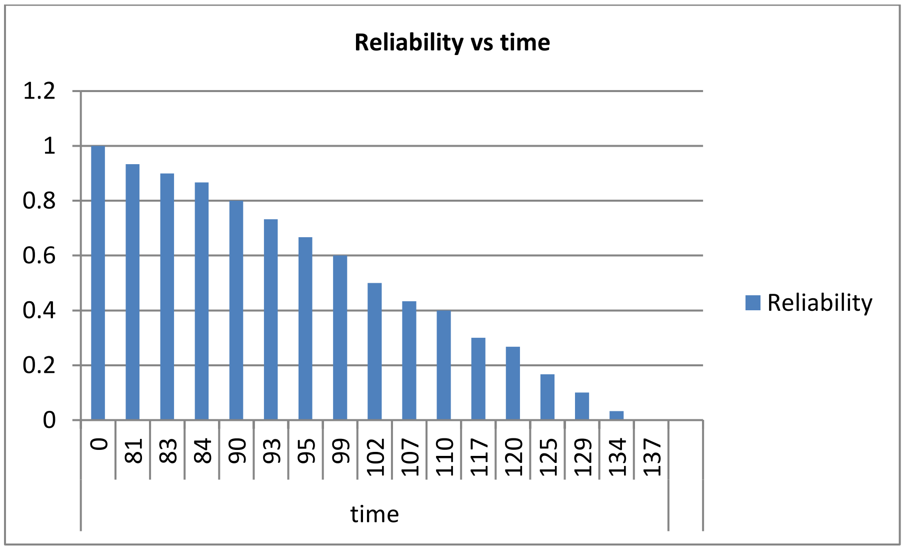

Figure 3. Reliability vs. time graph

Column 7 of Table 2 is the calculated reliability. Figure 2 and 3 respectively show the growth/decay of failure rate and reliability. Figure 2 shows the failure rate vs. time graph based on proposed approach. As can be seen from Figure 2, the failure rate is increasing with time. The reliability vs. time graph is shown in Figure 3. It shows that the reliability value decreases as time increases.

4. Conclusion

Reliability of a software or system is the probability of system to perform its functions adequately for the stated time period under specific environmental conditions. In case of component-based software development reliability estimation is a crucial factor. Existing reliability estimation models falls into two broad categories parametric and non-parametric models. Parametric models approximate the model parameters based on the assumptions of fundamental distributions. Non-parametric models enable parameter estimation of the software reliability growth models without any assumptions. We have proposed a novel non-parametric approach for survival analysis of components. Failure data is collectively based on this. We have calculated failure rate and reliability on the basis of this software. Failure rate increases with the time whereas reliability decreases with the time. Various authors proposed parametric approaches for estimating reliability of the CBS. Thus, we have tried to contribute a non-parametric approach.

Conflict of Interest

All authors have contributed equally in this work. The authors declare that there is no conflict of interest for this publication.

Acknowledgment

The authors would like to thank all participants in the proposal for their active and valuable responses.

References

Bowers, J., & Khorakian, A. (2014). Integrating risk management in the innovation project. *European Journal of Innovation Management*, *17*(1), 25-40.

Everett, W.W. (1999). Software component reliability analysis. In *Proceedings 1999 IEEE Symposium on Application-Specific Systems and Software Engineering and Technology. ASSET'99 (Cat. No. PR00122)* (pp. 204-211). IEEE. Richardson, TX, USA.

Gayen, T., & Misra, R.B. (2009). Reliability assessment of elementary COTS software component. *International Journal on Recent Trends Engineering, 1*(2), 196-200.

Gayen,T., & Misra, R.B. (2008). Reliability bounds prediction of COTS component based software application. *International Journal of Computer Science and Network Security, 8*(12), 219-228.

Gokhale, S.S. (2007). Architecture-based software reliability analysis: overview and limitations. *IEEE Transactions on Dependable and Secure Computing, 4*(1), 32-40.

Gokhale, S.S., Lyu, M.R., & Trivedi, K.S. (1998, November). Reliability simulation of component-based software systems. In *Proceedings Ninth International Symposium on Software Reliability Engineering (Cat. No. 98TB100257)* (pp. 192-201). IEEE. Paderborn, Germany.

Goševa-Popstojanova, K., & Trivedi, K.S. (2003). Architecture-based approaches to software reliability prediction. *Computers & Mathematics with Applications*, *46*(7), 1023-1036.

Goswami, V. & Acharya, Y.B. (2009). Method for reliability estimation of COTS components based software systems. *International Symposium on Software Reliability Engineering* (*ISSRE2009*). Mysuru, India

Isaac, I. (1995). Training in risk management. *International Journal of Project Management*, *13*(4), 225-229.

Karunanithi, N., Whitley, D., & Malaiya, Y.K. (1992). Using neural networks in reliability prediction. *IEEE Software*, *9*(4), 53-59.

Lakshmanan, I., & Ramasamy, S. (2015). An artificial neural-network approach to software reliability growth modeling. *Procedia Computer Science*, *57*, 695-702.

Lo, J.H. (2010, June). Early software reliability prediction based on support vector machines with genetic algorithms. In *2010 5th IEEE Conference on Industrial Electronics and Applications* (pp. 2221-2226). IEEE. Taichung, Taiwan.

Nautiyal, L., & Preeti. (2016), Evaluating and certifying component-based software using weighted assignment technique. *International Journal of Hybrid Information Technology*, *9*(1), 241-252

Singh, H., Cortellessa, V., Cukic, B., Gunel, E., & Bharadwaj, V. (2001, November). A Bayesian approach to reliability prediction and assessment of component based systems. In *Proceedings 12th International Symposium on Software Reliability Engineering* (pp. 12-21). IEEE. Hong Kong, China.

Su, Y.S., & Huang, C.Y. (2007). Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models. *Journal of Systems and Software*, *80*(4), 606-615.

Tyagi, K., & Sharma, A. (2012). A rule-based approach for estimating the reliability of component-based systems. *Advances in Engineering Software*, *54*, 24-29.

Wang, D., & Huang, N. (2008, October). Reliability analysis of component-based software based on rewrite logic. In *2008 12th IEEE International Workshop on Future Trends of Distributed Computing Systems* (pp. 126-132). IEEE. Kunming, China.

Weiss, S.N., & Weyuker, E.J. (1988). An extended domain-based model of software reliability. *IEEE Transactions on Software Engineering*, *14*(10), 1512 – 1524.

Whittaker, J.A., Rekab, K., & Thomason, M.G. (2000). A Markov chain model for predicting the reliability of multi-build software. *Information and Software Technology*, *42*(12), 889-894.

Yacoub, S., Cukic, B., & Ammar, H.H. (2004). A scenario-based reliability analysis approach for component-based software. *IEEE Transactions on Reliability*, *53*(4), 465-480.

Yacoub, S.M., & Ammar, H.H. (2002). A methodology for architecture-level reliability risk analysis. *IEEE Transactions on Software Engineering*, *28*(6), 529-547.

Zhang, F., Zhou, X., Dong, Y., & Chen, J. (2009, May). Consider of fault propagation in architecture-based software reliability analysis. In *2009 IEEE/ACS International Conference on Computer Systems and Applications* (pp. 783-786). IEEE. Rabat, Morocco.