International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 635–653, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-051

A Method for Considering Error Propagation in Reliability Estimation of Component-Based Software Systems

Preeti Malik[a*], Lata Nautiyal[b], Mangey Ram[c]
[a,b,c]Department of Computer Science and Engineering
[c]Department of Mathematics
Graphic Era Deemed to be University, Dehradun, Uttarakhand, India
[*]*Corresponding author*: preetishivach2009@gmail.com

(Received September 29, 2018; Accepted February 11, 2019)

Abstract
Component-based software engineering has proved itself as a strong pillar in software engineering community. Large number of factors are involved in the success of software product developed using Component-based software engineering, for instance, security, reliability, quality, safety, and testability. As the component-based software is made up of large number of components put together, therefore components' reliability influence the reliability of the complete software. Numerous models and principles have been established to estimate the reliability of software by applying information regarding architecture, quality and security attributes of the software. Many researchers overlooked a major factor while estimating reliability of component-based software and that is *error-propagation.* Although it can be a case that the components are not dependent on each other and they are supplemented with the wrappers. However it is not true for many component-based applications. In this paper, a framework for reliability estimation has been proposed. In our proposal we have considered error propagation. We have analyzed the program structure and also presented how they are handled in estimation process. Further sensitivity analysis is done to identify the most sensitive component of the system. A numerical simulation is also presented for better understanding of the proposed framework.

Keywords- Component-based software, Reliability estimation, Error propagation, Path-based model, Transition probability, Sensitivity analysis.

Acronyms

CBS	Component-Based System/Software
CBSE	Component-Based Software Engineering
EPP	Error Propagation Probability
NHPP	Non-Homogeneous Poisson Process
SDLC	Software Development Life Cycle
UML	Unified Modeling Language
CDG	Component Dependency Graph
CASE	Component-Aided Software Engineering
SRGM	Software Reliability Growth Model
COTS	Component-off-the-shelf
RADL	Rich Architecture Definition Language
CRS	Component Reliability Specification
PACS	Personnel Access Control System

Notations

R_{Ci}	Reliability of i^{th} component
G	A Component flow graph, G= <C, T, P, R>
C	Set of components
T	Set of edges between components
P	Set of transition probability
R	Set of reliability of components

International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 635–653, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-051

C_i	A specific component
T_{ij}	Transition from i^{th} component to j^{th} component
M	Total number of components
P	Number of testing paths
p_k	A specific testing path
R_{pk}	Reliability of p_k^{th} testing path
n_i	Busy period of i^{th} component
$EP(X, Y)$	Error propagation between component X and Y
i, j	Index of components, i, j= 1, 2, 3…..M
l	Number of iterations in a loop structure
$E(X,Y)$	Error propagation probability
$T(X,Y)$	Transition probability
Θ_j	Failure probability of j^{th} component
$P_Y(x)$	Probability of component Y in state x (S_Y is the set of states of component Y)
$P_{x\to y}[F_x^{-1}(y)]$	Probability of message transmission that causes component to go to state y
$P_{X\to Y}[v]$	Probability of sending a message from component X to Y
R_{PCi}	Reliability of component C_i while considering error propagation
$T_{ch,Ri}$	Relative change of system reliability (when reliability of a component is changed with the factor *ch)*
$S_{ch,Ri}$	Sensitivity of system (when reliability of a component is changed with the factor *ch*)
Ch	Change in reliability of a component

1. Introduction

"Simplicity is prerequisite for reliability"—Edsger W. Dijkstra

Use of modular design in software development process is increasing as the complexity and size of the software is increasing (Misra and Sharma, 1991; Goševa-Popstojanova and Trivedi, 2001; Gokhale, 2007). Component-based development have tremendously gained acceptance in the world of software development. The ultimate aim is to design independent components which are part of the software system and collaborate with the available system components (Heineman and Councill, 2001). The fundamental benefit behind using the component-based software development approach includes, high quality product in low cost using the minimal efforts and reduction in development time. These pre-tested components make them highly useful and suitable to use in large number of applications (D"Souza and Wills, 1997; Lau et al., 2006; Nautiyal et al., 2013). These all benefits achieved because a CBSE approach rely on predefined software components. Along with the above challenges, quality and reliability of components or modules may poses new challenges in front of developers. It is thus a requirement to have an improved technique for estimating reliability of the software at its early development phase (Musa et al., 1987; Xie, 1991; Lyu, 1996).

Numerous new models and procedures are proposed for effectively refining the estimation of reliability of CBS (Littlewood, 1979; Schneidewind and Keller, 1992; Pasquini et al., 1996; Gokhale and Trivedi, 1997; Gokhale et al., 1998; Wang et al., 1999; Pham, 2006; Malaiya et al., 2002; Huang et al., 2003; Meyer, 2003; Tian et al., 2004; Popic et al., 2005; Huang and Lin, 2006; Almering et al., 2007; Dubey et al., 2017). However, existing reliability prediction approaches for CBS lack in some of the areas. In short, these disadvantages are penalty of the assumption that components fail independently and each component failure leads to a system failure, which is common to most existing reliability models for CBS. Hence most essential issue faced in component-based developers' community is the reliability (Sitaraman and Weide, 1994; Garg et al., 2016). There are two main categories of reliability estimation processes, which are the system level and component level. Approaches of first category don't consider individual

components' information and treat system as a whole (Singh et al., 2001). In second category, the approaches use components' information as a major factor in reliability analysis.

It is ever believed that a faults in software and the errors resulted from that fault (in any of the component) are propagated to other components communicating or linked with that particular component, and this is the most prominent cause of the failure of the components. Therefore failures among the components are ever because of that communication or dependency of components on each other. Error propagation is an essentials element from the developer's point of view since the process of estimation is applied to locate the most affected components and to improve the reliability of the software system by the application of error detection and recovery mechanism. Although a large number of studies have been proposed by numerous authors for estimating error propagation among components of the system (Voas, 1997; Abdelmoez et al., 2004; Cortellessa and Grassi, 2007; Pham and Defago, 2013). However some authors do not consider it (Cheung, 1980; Reussner et al., 2003; Sharma and Trivedi, 2007; Cheung et al., 2008; Brosch et al., 2012; Nautiyal et al., 2014). They assume that the control transfer between components can be explained as a Markov process. This assumption denotes that execution of next components only depends on execution of present component and is independent of past although these assumptions don't hold for all the applications. Hence we have included error propagation in our approach. Three major factors which affect the system reliability (Cortellessa and Grassi, 2007):

- Failure probability of components
- Error propagation probability
- Path propagation probability

First one among these factors is component-level factors whereas remaining two are architecture-level factors. If we neglect the factor of error propagation in reliability estimation then this may result in over pessimistic analysis of reliability. Further, it may result in risk of redundant design and execution efforts to increase reliability and there is also risk of mistaken decision in component selection as well as architecture selection.

In this article, we have proposed framework for reliability estimation of CBSs. Here, we have considered a vital architectural feature that is generally ignored for keeping reliability model simple and this feature is EPP. It denotes the probability of occurrence of an error in one component and its propagation to another component. The objective of this research is to analyze the impression of error propagation on reliability of CBS. Based on the program structure, we have proposed methods to calculate path reliability. These path reliabilities are then used to estimate the system reliability. A numerical simulation of the proposed approach is also given.

Rest of the paper is organized as follows. Section 2 includes background study. Section 3 talks about the details of error propagation and also method for computing EPP. Section 4 includes the proposed framework. In Section 5, we have presented numerical simulation of proposed approach on PACS and section 6 we have sensitivity analysis of PACS for finding most sensitive component. At last, Section 7 concludes the findings.

2. Background Study

SRGM model treat the system as a black-box while estimating reliability. In our approach, we have considered our system as a composition of software components. Some literature provide in

International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 635–653, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-051

depth survey of this field (Immonen and Niemelä, 2007). CBS differs from conventional systems as the system is broken up into separate logical units that are integrated together to form a complete software system. These units are COTS software products that are already developed. Hence, the reliability of the whole system is considered to be a function of the reliability of the components integrated to make the whole software system and the operational profile of the system. Different categories of CBS reliability models, as suggested by (Goševa-Popstojanova and Trivedi, 2001) are:

- State-Based Models
- Path-Based Models
- Additive Models

We have used path-based model hence we will discuss only path based approaches. In these approaches, we analyze various execution paths along with their execution frequency. These models require complete source code of the CBS so as to create information about paths. This information is then joined with the failure behavior for estimating reliability. Shooman model (Shooman, 1973) and Yacoub et al. (2004) are the example of path-based models. Shooman model is the first and foremost model that uses path-based approach for reliability estimation of a system. Shooman assumes that the execution paths i and their frequencies f_i are known already and failure probability of each path is denoted by q_i. So the probability of system failure can be modeled as:

$$q_0 = \sum_{i=1}^{n} f_i * q_i$$

where, n is the component count in the system (Shooman, 1973).

Yacoub provided a way for before time prediction of reliability. This strategy is strictly based on execution scenario (Yacoub et al., 2004). A scenario can be defined be defined as a set of interactions among components executed by exact input. Sequence diagrams are used by Yacoub to define a scenario and these sequence diagrams are same as sequence diagram of UML. Average execution time of components, average execution time of a scenario, and interaction among different components information is provided by these diagrams. A CDG is constructed by using the scenarios. A CDG is a modified control flow graph. Algorithm is then applied to CDG to estimate the reliability of CBS. Cheung (Cheung, 1980) showed the flow of control between components of the system as a Discrete Markov Chain Model. Some authors extended his proposal to various architectural styles (Wang et al., 1999) and analyze performance (Sharma and Trivedi, 2007) but they also don't consider error propagation. Lipton (Lipton and Gokhale, 2008) is one of those models who has extended Cheung's model. Lipton considered interface failure and connection failure too.

Reussner also used the same approach as Cheung's but the difference is that it is based on RADL (Reussner et al., 2003). Reussner approach is extended by (Brosch et al., 2012). Brosch discussed the influence of system usage profile and operational environment on system reliability. These approaches again don't take error propagation into account. (Sato and Trivedi, 2007) merge resource availability model and system model but again pays no attention to error propagation. (Grassi, 2005) proposed an approach that makes use of recursively composed services. (Zheng

International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 635–653, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-051

and Lyu, 2010) discussed sequential, loop and parallel structure of composite services. Both of these approaches ignore error propagation.

The approaches given by Cortellessa et al. (2002); Goseva-Popstojanova et al. (2003) utilized UML diagrams for reliability estimation. Rodrigues et al. (2005) proposed an approach, also don't consider error propagation. This approach is based on message sequence chart. Popic et al. (2005) and Cortellessa and Grassi (2007) considered error propagation in reliability estimation process. Authors in Filieri et al. (2010), Mohamed and Zulkernine (2008) considered multiple failure types. Most of these methods ignore the concept of error propagation for parallel and fault tolerance execution model which are often used by modern software systems.

3. Computing Error Propagation Probability

Nassar defined error propagation as given in Eq. (1) (Nassar et al., 2002). EPP represents the probability that a fault occurred in component X will result in erroneous output from component Y. Error propagation is denoted by a MxM matrix if there are M components in the system where EP(X,X) is equal to 1 because if a component fails then it is assumed to remain fail. Nassar suggested error propagation between component X and Y can be defines as:

$$EP(X,Y)=\frac{1-\sum_{x\in S_Y} P_Y(x)\sum_{y\in S_Y} P_{X\to Y}[F_x^{-1}(y)]^2}{1-\sum_{v\in V_{X\to Y}} P_{X\to Y}[v]^2} \quad (1)$$

We consider error propagation an unconditional event i.e. error is propagated from component X to Y without being conditioned by the occurrence of a message from component X to Y. This unconditional error propagation probability is denoted by E(X, Y) and is calculated from *Transition Probability* (T(X, Y)) by using the formula:

$$E(X, Y)=EP(X, Y)*T(X, Y) \quad (2)$$

T(X, Y)= (number of message between component X and Y) / (total number of observed messages in the system).

By introducing error propagation in reliability estimation the reliability of components will be modified. The probability of component C_i (without failure dependence) is

$$P(\text{success of } C_i \mid \text{without failure dependence})= (R_{Ci})^{n_i} \quad (3)$$

where, n_i is the busy period of that component. Now remove the assumption of failure independence. The component get executed successfully if no fault is propagated from other components and also it does not counter its own fault i.e.

$$P(\text{success of } C_i)= P(\text{success of } C_i \mid \text{without failure dependence})* P(\text{no\_error\_propagated}) \quad (4)$$

EPP from component j to i can be given by

$$P(\text{error\_prop } j\to i)=E(j, i)\ \Theta_j \quad (5)$$

International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 635–653, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-051

$$P(no\_error\_prop\ j \rightarrow i) = 1 - E(j,i)\,\Theta_j \quad (6)$$

$$R_{PCi} = P(success\ of\ C_i) = (R_{Ci})^{n_i} * \prod_{j=1}^{M} (1 - E(j,i)\theta_j) \quad (7)$$

4. Proposed Framework for Reliability Estimation

4.1 Definitions

***Definition* 1.** A CBS can be represented as a CDG. Let G be the CDG which can be defined as G=<C, T, P, R>, where C is the set of components, T is set of edges between any two components and P is the set of transition probabilities and R is the set of reliability of components.

***Definition* 2.** A directed edge from C_i to C_j is T_{ij}. This edge denotes transition from i^{th} component to j^{th} component. If $T_{ij}=0$, the nodes are not connected directly.

***Definition* 3.** Testing path is a sequence of components travelled during execution. Reliability of a testing path is the result of contribution of those components. The possible path of execution can be identified by traversing the nodes starting from entry node to exit node.

Assumptions

- **a.** For every component (C_i) of the system, the reliability (R_{Ci}) of the component is already known in prior.
- **b.** Failure rates for component and connectors are already given.
- **c.** EPP is already given. We have assumed that we have EPP of components in advance.
- **d.** Component failure is constant. A component is presumed to show similar failure behavior all the time it is invoked.
- **e.** Failure of different components is independent.

4.2 Proposed Framework

Following Figure 1 depicts the process of reliability estimation. In first step, developers of components make available their CRS. This specification includes the information like failure probabilities, error propagation probability, and transition probability etc. Method for finding these values can be found in (Cheung, 1980; Hiller et al., 2004; Brosch et al., 2012). These methods are beyond the scope of this paper. Next step is to construct component CDG. There are various CASE tools available in the market to construct graphical representation of the system like entity-relationship diagram, control flow diagram. By analyzing control flow diagram we can create CDG.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 635–653, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-051

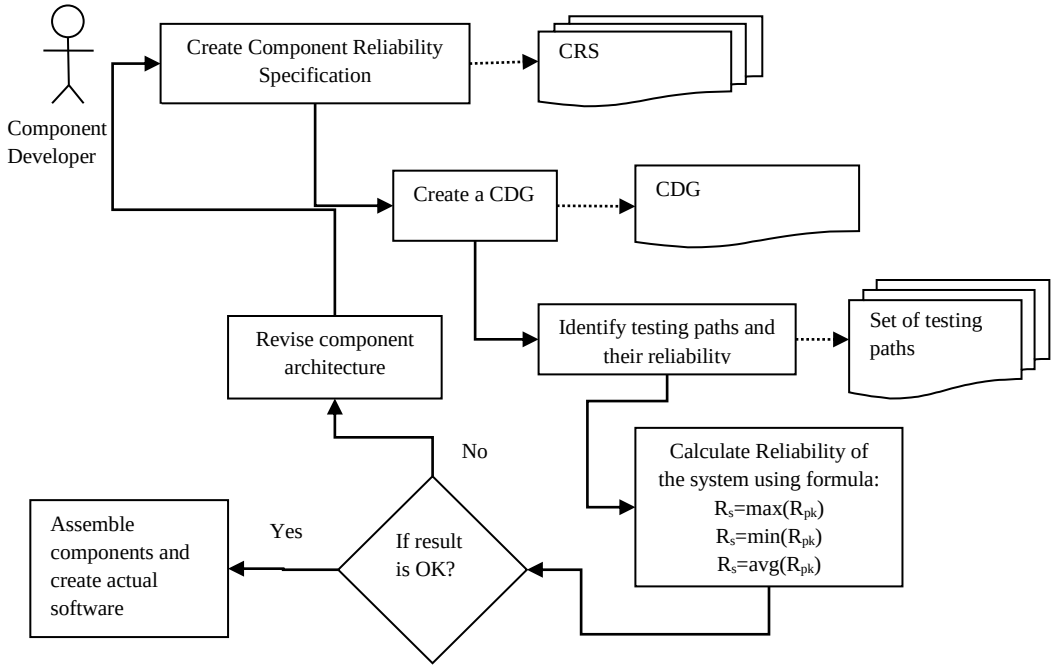

Figure 1. Proposed framework

Third step is to identify all the possible test paths. These testing paths may vary according to the program structure. We have taken into account three program structures: sequential structure, branch structure and loop structure (discussed in detail in following section). In general, if there are p_k different execution paths in system then the reliability of a CBS can be given by following three different formulae

$R_s=max(R_{pk})$
$R_s=min(R_{pk})$
$R_s=avg(R_{pk})$

where R_{pk} is the reliability of pk^{th} path. First one is an optimistic estimate; second one is conservative estimate and third is a moderate estimate (Krishnamurthy and Mathur, 1997; Schneidewind, 2009). From the last two estimates, we can observe that selection of testing path greatly affect the accuracy of reliability (Chen et al., 2001; Myers et al., 2011; Jorgensen, 2008). If acceptable level of reliability is reached then we can use those components and built our real system otherwise we have to change the component or their usage profile etc. and then the whole process is repeated.

4.3 Introducing Program Structure

This section is dedicated to extending the proposed reliability analysis to include program structure while estimating reliability of the CBS. We have considered three program structures; sequential structure, branch structure and loop structure.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 635–653, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-051

Sequential Structure: Let us assume that the system execution is following a sequential path which consists of N nodes or components. C_1 is the entry component and C_N is the exit component. Then the path reliability can be given by

$$R_{p_k} = R_1 * \prod_{i=2}^{N} R_{PC_i} \tag{8}$$

where, $R_{PCi}=P(success\ of\ C_i) =(R_{Ci})^{n_i} * \prod_{j=1}^{N}(1-E(j,i)\theta_j)$.

Proof: The sequential structure is executed in a fixed order (Figure 2). As we have considered error propagation, component's reliability can be calculated as (from section 3)

$$R_{PC_i} = (R_i)^{n_i} * \prod_{j=1}^{N}(1-E(j,i)\theta_j)$$

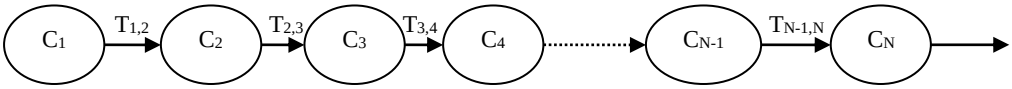

Figure 2. Sequential structure

Component C_1 is always executed only once when the system is entered. So reliability of sequential path can be measured as:

$$R_{p_k} = R_1 * \prod_{i=2}^{N} R_{PC_i}\ .$$

Branch Structure: Let us assume that the system execution contains N-2 branches and each branch is independent of each other. Then the path reliability can be given by

$$R_{p_k} = R_1 * \prod_{i=2}^{N-1} R_{PC_i}^{n_i} * R_{PC_M}{}^{N-2} \tag{9}$$

Proof: To verify the above formulae, consider the system structure as shown in Figure 3. Each branch in the system can be considered a sequential path so there are N-2 paths in the system and output node must be visited during each path. Using similar calculation of sequential structure:

$$\begin{aligned} R_{p_k} &= R_1 * (R_2 * R_3 .... R_N) * (R_i * R_{i+1} .... R_N) \\ &= R_1 * \prod_{i=2}^{N-1} R_{PC_i} * R_{PC_N}{}^{N-2} \end{aligned} \tag{10}$$

International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 635–653, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-051

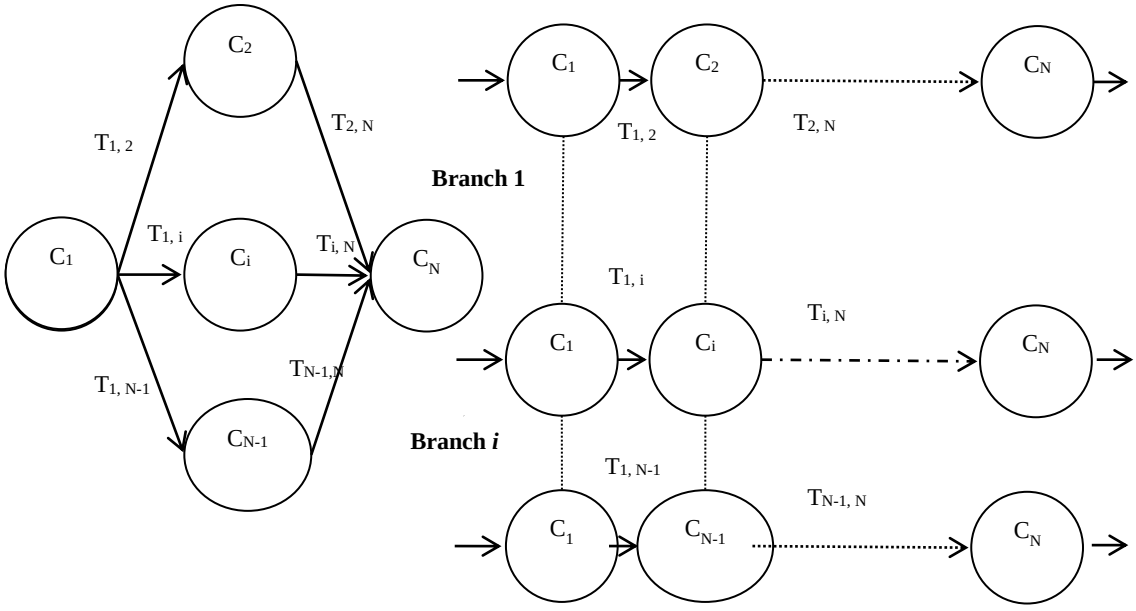

Figure 3. Branch structure and expansion

Loop Structure: Third program structure we have considered is loop structure. Assume testing path is a loop structure of l iterations and there are 5 nodes in a single iteration where C_1 is the entry node and C_4 is the exit node. Hence reliability can be defined as

$$R_{p_k} = R_1 * R_2^{\frac{l(l+1)}{2}} * R_3^{\frac{l(l+1)}{2}} * R_4^{\,l} \tag{11}$$

Proof: In loop structure component C_2, C_3 are executing repeatedly (Figure 4). Similar to sequential and branch structure we have calculated the reliability in case of loop structure. As can be seen from the Figure 6, in first iteration of the loop component C_2 and C_3 are used only once, in second iteration they used two time and this busy period increase as the loop iteration increases. So for l iterations busy period of components C_2 and C_3 forms arithmetic progression whose first term (a) is 1 and difference (d) is 1. Therefore the sum of arithmetic progression up to l terms

$S= n/2\{a+nd\}= l/2\{1+l*1\}=l(l+1)/2$

International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 635–653, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-051

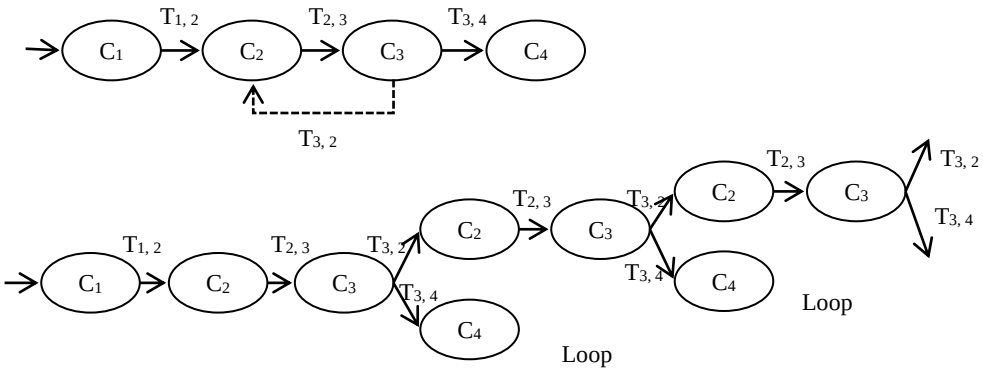

Figure 4. Loop structure and expansion

$$
\begin{aligned}
R_{p_k} = & \left(\underbrace{R_1 * R_2 * R_3 * R_4}_{iteration\ 1} \right) * \\
& \left(\underbrace{(R_2 * R_3) * (R_2 * R_3) * R_4}_{iteration\ 2} \right) * \\
& \left(\underbrace{(R_2 * R_3) * (R_2 * R_3) * (R_2 * R_3) * R_4}_{iteration\ 2} \right) * \\
& \ldots\ldots \left(\underbrace{\{(R_2 * R_3) * (R_2 * R_3) * (R_2 * R_3) \ldots\ldots l\ times\} * R_4}_{iteration\ 1} \right) \\
= & \ R_1 * ((R_2 * R_3)^1 * R_4) * ((R_2 * R_3)^2 * R_4) * ((R_2 * R_3)^3 * R_4) * \ldots\ldots ((R_2 * R_3)^l * R_4) \\
= & \ R_1 * R_2^{1+2+3\ldots\ldots l} * R_3^{1+2+3\ldots\ldots l} * R_4^{1+1+1\ldots\ldots l} \\
= & \ R_1 * R_2^{\frac{l(l+1)}{2}} * R_3^{\frac{l(l+1)}{2}} * R_4^{\,l}.
\end{aligned}
$$

5 Numerical Simulation

We have applied the proposed approach on PACS. System description is given in National Security Agency (2003). PACS consists of 5 components. And their reliabilities are R_{C1}=0.998, R_{C2}=0.995, R_{C3}=0.998, R_{C4}=0.994, R_{C5}=0.994. Failure probability and EPP are taken from Popic et al. (2005). According to Lil et al. (2004) , they have implemented the project under two different configuration and reliabilities are found 91.6 and 99.85. Data is as follows (Table 1 and Table 2).

International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 635–653, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-051

Table 1. EPP matrix

	C1	C2	C3	C4	C5
C1	0	0.425169	0	0	0
C2	0.804878	0	0.02439	0.08338	0.073165
C3	0	0.687124	0	0	0
C4	0	0	0	0	0
C5	0	0	0	0	0

Table 2**.** Information record

Busy Period	Failure Probability =(1-Reliability)	Scenario1	Scenario 2	Scenario 3
C1	0.002	4	5	24
C2	0.005	1	1	4
C3	0.002	0	0	1
C4	0.006	1	1	3
C5	0.006	1	1	0

5.1 Effect of Error Propagation on System Reliability

5.1.1 System Reliability without Error Propagation

For Scenario 1:

$R_{p1}=(R_{C1})^4*(R_{C2})^1*(R_{C3})^0*(R_{C4})^1*(R_{C5})^1$

$=(0.998)^4*(0.995)^1*(0.998)^0*(0.994)^1*(0.994)^1$

$=0.97525462.$

For Scenario 2:

$R_{p2}=(R_{C1})^5*(R_{C2})^1*(R_{C3})^0*(R_{C4})^1*(R_{C5})^1$

$=(0.998)^5*(0.995)^1*(0.998)^0*(0.994)^1*(0.994)^1$

$=0.97330411.$

For Scenario 3:

$R_{p3}=(R_{C1})^{24}*(R_{C2})^4*(R_{C3})^1*(R_{C4})^3*(R_{C5})^0$

$=(0.998)^{24}*(0.995)^4*(0.998)^1*(0.994)^3*(0.994)^0$

$=0.91561945.$

By taking conservative approach (from section 4.2),

$R_s=\min(R_{p1},R_{p2},R_{p3})=\min(0.97525462,0.97330411,0.91561945)=0.91561945$

By taking optimistic approach,

$R_s=\max(R_{p1},R_{p2},R_{p3})=\max(0.97525462,0.97330411,0.91561945)=0.97330411$

International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 635–653, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-051

By taking moderate estimate,

R_s=avg(R_{p1},R_{p2},R_{p3})= (0.97525462+0.97330411+0.91561945)/3=0.95472606.

5.1.2 System Reliability with Error Propagation

For Scenario 1:

R_{PC1}=*P(success of* C_1 *under failure dependence)*= $(0.998)^4$(1-0.425169*0.002)

=$(0.998)^4$*(1-0.000850338)=0.99202397*0.99914966=0.99118041

R_{PC2}=(0.995)(1-0.804878*0.005)(1-0.02439*0.002)(1-0.08338*0.006)(1-0.073165*0.006)

=0.995(1-0.00402439)(1-0.00004878)(1-0.00050028)(1-0.00043899)

=0.995(0.99597561)(0.99995122)(0.99949972)(0.99956101)=0.995*0.99498929

=0.99001684.

R_{PC3}=$(0.998)^0$(1-0.687124*0.005)=1-0.00343562=0.99656438

R_{PC4}=$(0.994)^1$=0.994.

R_{PC5}=0.994.

So, testing path p_1 reliability, $R_{p1=}$ 0.96621421.

For Scenario 2:

R_{PC1}=$(0.998)^5$(1-0.425169*0.002)= 0.98919805.

R_{PC2}=(0.995)(1-0.804878*0.005)(1-0.02439*0.002)(1-0.08338*0.006)(1-0.073165*0.006)

=0.995(1-0.00402439)(1-0.00004878)(1-0.00050028)(1-0.00043899)

=0.995(0.99597561)(0.99995122)(0.99949972)(0.99956101)

=0.99001684.

R_{PC3}=$(0.998)^0$(1-0.687124*0.005)=1-0.00343562=0.99656438.

R_{PC4}=$(0.994)^1$=0.994.

R_{PC5}=0.994.

So, testing path p_2 reliability, R_{p2}=0.96428178.

For Scenario 3:

R_{PC1}=$(0.998)^{24}$(1-0.425169*0.002)

=0.95308798(1-0.000850338)=0.95308798*0.99914966=0.95227753

R_{PC2}=$(0.995)^4$(1-0.804878*0.005)(1-0.02439*0.002)(1-0.08338*0.006)(1-0.073165*0.006)

International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 635–653, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-051

=0.97524072

$R_{PC3}=(0.998)^1(1-0.687124*0.005)=0.998*(1-0.00343562)=0.998*0.99656438=0.99457125$

$R_{PC4}=(0.994)^3=0.98210778$

$R_{PC4}=0.994.$

So, testing path p_3 reliability, $R_{p3}=0.90168906$.

By taking conservative approach,

$R_s=\min(R_{p1},R_{p2},R_{p3})=\min(0.96621421,0.96428178,0.90168906)=0.90168906.$

By taking optimistic approach,

$R_s=\max(R_{p1},R_{p2},R_{p3})=\max(0.96621421,0.96428178,0.90168906)=0.96621421.$

By taking moderate estimate,

$R_s=\text{avg}(R_{p1},R_{p2},R_{p3})=(0.96621421+0.96428178+0.90168906)/3=0.94406168.$

Following Figure 5 shows the effect of considering error propagation in reliability estimation. As we can, reliability of the system decreases when we took error propagation into account.

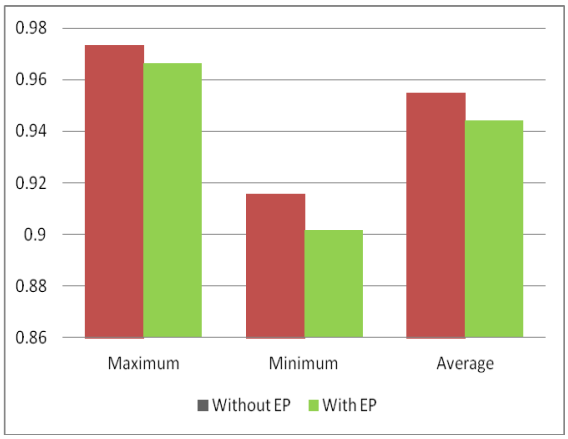

Figure 5. Effect of considering error propagation

6 Finding the Most Sensitive Component

Sensitivity is a measure that is used to analyze the effect of change in reliability of a component on overall system reliability. Most sensitive component have a higher effect on system reliability, hence we can improve system reliability by improving that particular component. In sensitivity

International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 635–653, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-051

analysis, we have to find such component which is more important than others. According to (Lo et al., 2003), we are concerned for the following condition

$$\left|\frac{\partial R_s}{\partial R_i}\right| \le \left|\frac{\partial R_s}{\partial R_j}\right|, \text{ for all j=1, 2, ...M} \tag{12}$$

We can define $T_{ch,\,Ri}$ and $S_{ch,Ri}$ as follows,

$$T_{ch,\,R_i} = \frac{\left|R_S(R_1 \ldots R_i + ch, \ldots R_M) - R_S(R_1 \ldots R_M)\right|}{R_S(R_1 \ldots R_M)} 100\% \tag{13}$$

$$S_{ch,R_i} = \frac{T_{ch,R_i}}{ch} 100\% \tag{14}$$

Therefore, i[th] component is said to be the most sensitive component if following inequality is satisfied:

$$S_{ch,R_i} \ge S_{ch,R_j}, \textit{ for all } \text{j=1,2,3....M} \tag{15}$$

Table 3. Sensitivity analysis (Scenario 1)

		Change in Reliability								
		-0.08	-0.06	-0.04	-0.02	0	0.02	0.04	0.06	0.08
Sensitivity	C1	-0.396008	-0.3972015	-0.3983982	-0.3995988	0	0.388426	0.39	0.4044367	0.40564675
	C2	-0.1005337	-0.1005441	-0.1005652	-0.1006288	0	0.1003763	0.10043918	0.1004603	0.10047074
	C3	0	0	0	0	0	0	0	0	0
	C4	-0.1006038	-0.1006036	-0.1006035	-0.1006034	0	0.1006035	0.10060372	0.10060363	0.1006036
	C5	-0.1006038	-0.1006036	-0.1006035	-0.1006034	0	0.1006035	0.10060372	0.10060363	0.1006036

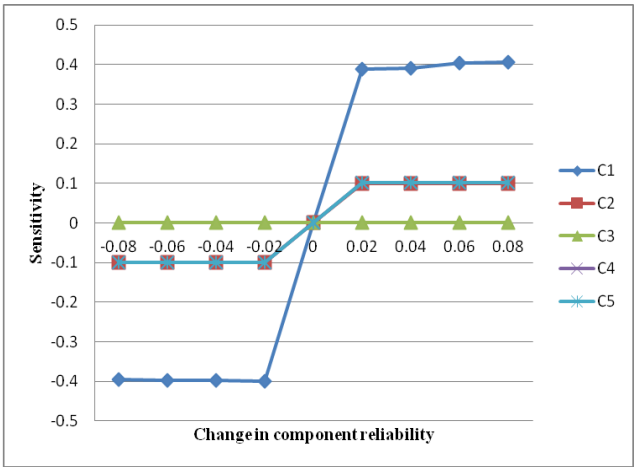

Figure 6. Sensitivity analysis (Scenario 1)

We have done sensitivity analysis of scenario 1 only, and found that C_1 is the most sensitive component and C_3 does not affect system reliability as C_3 is not participating in that particular scenario. Sensitivity curves of component C_4 and C_5 are overlapping it indicates that they equally sensitive because they have same reliability and same busy period in that particular scenario. Results of sensitivity analysis are presented in Table 3 and Figure 6.

7. Conclusion

In this article, we have proposed a framework for estimating reliability of CBS while considering EPP between components of the system. Many researchers overlooked this major factor**.** Although it can be the case if the components are not dependent on each other but not true for many CBS. We have analyzed the program structure in estimation process. Three program structures are taken into account viz; sequential structure, branch structure and loop structure. Further, the proposed approach is applied to a system (PACS) for three scenarios. Sensitivity analysis is done to identify the most critical component, which affects system reliability more than other components.

Open issues for future research are; the proposed approach demand significant efforts for estimation process. Hence an automated tool for estimation can be found like; UML for example, for calculating error propagation matrix etc. Moreover, concurrent errors in multiple components can also be considered in near future. Further we have assumed that component failure is constant, in future we can relax this assumption and improve our approach in this direction.

Conflict of Interest

The author(s) confirm that this article contents have no conflict of interest.

Acknowledgement

The authors would like to express their sincere thanks to the referee and for their valuable suggestions towards to the improvement of the paper.

References

Abdelmoez, W., Nassar, D.M., Shereshevsky, M., Gradetsky, N., Gunnalan, R., Ammar, H.H., Yu, B, & Mili, A. (2004, September). Error propagation in software architectures. In *Software Metrics, 2004. Proceedings. 10th International Symposium on* (pp. 384-393). IEEE.

Almering, V., van Genuchten, M., Cloudt, G., & Sonnemans, P.J. (2007). Using software reliability growth models in practice. *IEEE Software*, *24*(6), 82-88

Brosch, F., Koziolek, H., Buhnova, B., & Reussner, R. (2012). Architecture-based reliability prediction with the palladio component model. *IEEE Transactions on Software Engineering*, *38*(6), 1319-1339.

Chen, M.H., Lyu, M.R., & Wong, W.E. (2001). Effect of code coverage on software reliability measurement. *IEEE Transactions on Reliability*, *50*(2), 165-170.

Cheung, L., Roshandel, R., Medvidovic, N., & Golubchik, L. (2008). Early prediction of software component reliability. In *Proceedings of the 13th International Conference on Software Engineering - ICSE '08* (p. 111-120). Leipzig, Germany: ACM Press.

Cheung, R.C. (1980). A user-oriented software reliability model. *IEEE Transactions on Software Engineering*, *SE-6*(2), 118–125.

Cortellessa, V., & Grassi, V. (2007). A *modeling approach to analyze the impact of error propagation on reliability of component-based systems*. In Schmidt, H.W., Crnkovic, I., Heineman, G.T. & Stafford, J.A. (Eds.), *Component-based software engineering* (pp. 140–156). Springer Berlin Heidelberg.

Cortellessa, V., Singh, H., & Cukic, B. (2002). Early reliability assessment of UML based software models. In *Proceedings of the 3rd International Workshop on Software and Performance* (pp. 302–309). New York, NY, USA: ACM.

D"Souza D.F., & Wills A.C. (1997). *Objects, components, and frameworks with UML: the catalysis(SM) approach*. Addison-Wesley. Retrieved from https://www.pearson.com/us/higher-education/program/D-Souza-Objects-Components-and-Frameworks-with-UML-The-Catalysis-SM-Approach/PGM93470.html

Dubey, S.K., Singh, S., & Chaudhary, H. (2017). Evaluation of reliability of critical software system using fuzzy approach. *International Journal of System Assurance Engineering and Management*, *8*(2), 1327–1335.

Filieri, A., Ghezzi, C., Grassi, V., & Mirandola, R. (2010). *Reliability analysis of component-based systems with multiple failure modes*. In Grunske, L., Reussner, R. & Plasil, F. (Eds.), *Component-Based Software Engineering* (pp. 1–20). Springer Berlin Heidelberg.

Garg, N., Nautiyal, L., & Preeti. (2016). Limiting the reliability of component based software system. *International Journal of Computer Science Engineering*, *5*(5), 216–224.

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., & Trivedi, K.S. (1997). Structure-based software reliability prediction. In *In Proc. of Fifth Intl. Conference on Advanced Computing (ADCOMP '97* (pp. 447–452), Chennai, India.

Gokhale, S.S., Wong, W.E., Trivedi, K.S., & Horgan, J.R. (1998, September). An analytical approach to architecture-based software reliability prediction. In *Computer Performance and Dependability Symposium, 1998. IPDS'98. Proceedings. IEEE International* (pp. 13-22). IEEE.

Goševa-Popstojanova, K., & Trivedi, K.S. (2001). Architecture-based approach to reliability assessment of software systems. *Performance Evaluation*, *45*(2-3), 179-204.

Goseva-Popstojanova, K., Hassan, A., Guedem, A., Abdelmoez, W., Nassar, D.E.M., Ammar, H., & Mili, A. (2003). Architectural-level risk analysis using UML. *IEEE Transactions on Software Engineering*, *29*(10), 946-960.

Grassi, V. (2005). Architecture-based reliability prediction for service-oriented computing. In de Lemos, R., Gacek, C., & Romanovsky, A. (Eds.), *Architecting Dependable Systems III* (pp. 279–299). Springer Berlin Heidelberg.

Heineman, G.T., & Councill, W.T. (Eds.). (2001). *Component-based software engineering: putting the pieces together*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Hiller, M., Jhumka, A., & Suri, N. (2004). EPIC: profiling the propagation and effect of data errors in software. *IEEE Transactions on Computers*, *53*(5), 512–530.

Huang, C.Y., Lyu, M.R., & Kuo, S.Y. (2003). A unified scheme of some nonhomogenous poisson process models for software reliability estimation. *IEEE Transactions on Software Engineering*, *29*(3), 261-269.

Huang, C.-Y., & Lin, C.-T. (2006). Software reliability analysis by considering fault dependency and debugging time lag. *IEEE Transactions on Reliability*, *55*(3), 436–450.

Immonen, A., & Niemelä, E. (2007). Survey of reliability and availability prediction methods from the viewpoint of software architecture. *Software & Systems Modeling*, *7*(1), 49.

Jorgensen, P.C. (2008). *Software testing: a Craftsman's approach*, Third Edition (3rd ed.). Auerbach. Retrieved from https://www.crcpress.com/Software-Testing-A-Craftsmans-Approach-Third-Edition/Jorgensen/p/book/9781439889510.

Krishnamurthy, S., & Mathur, A.P. (1997, November). On the estimation of reliability of a software system using reliabilities of its components. In *Software Reliability Engineering, 1997. Proceedings., The Eighth International Symposium on* (pp. 146-155). IEEE.

Lau, K.K., Ornaghi, M., & Wang, Z. (2005, November). A software component model and its preliminary formalisation. In *International Symposium on Formal Methods for Components and Objects* (pp. 1-21). Springer, Berlin, Heidelberg.

Lil, M., Wei, Y., Desovski, D., Nejad, H., Ghose, S., Cukic, B., & Smidts, C. (2004, November). Validation of a methodology for assessing software reliability. In *Software Reliability Engineering, 2004. ISSRE 2004. 15th International Symposium on* (pp. 66-76). IEEE.

Lipton, M.W., & Gokhale, S.S. (2008). Heuristic component placement for maximizing software reliability. In H. Pham (Ed.), *Recent Advances in Reliability and Quality in Design* (pp. 309–330). London: Springer London.

Littlewood, B. (1979). Software reliability model for modular program structure. *IEEE Transactions on Reliability*, *R-28*(3), 241–246.

Lo, J.H., Huang, C.Y., Kuo, S.Y., & Lyu, M.R. (2003, November). Sensitivity analysis of software reliability for component-based software applications. In *Computer Software and Applications Conference, 2003. COMPSAC 2003. Proceedings. 27th Annual International* (pp. 500-505). IEEE.

Lyu, M. R. (1996). *Handbook of software reliability engineering* (Vol. 222). CA: IEEE Computer Society Press. New York: McGraw-Hill.

Malaiya, Y.K., Li, M.N., Bieman, J.M., & Karcich, R. (2002). Software reliability growth with test coverage. *IEEE Transactions on Reliability*, *51*(4), 420–426.

Meyer, B. (2003). The grand challenge of trusted components. In *Proceedings of the 25th International Conference on Software Engineering* (pp. 660–667). Washington, DC, USA: IEEE Computer Society.

Misra, K.B., & Sharma, U. (1991). An efficient algorithm to solve integer-programming problems arising in system-reliability design. *IEEE Transactions on Reliability*, *40*(1), 81-91.

Mohamed, A., & Zulkernine, M. (2008, August). On failure propagation in component-based software systems. In *The Eighth International Conference on Quality Software* (pp. 402-411). IEEE.

Musa, J.D., Iannino, A., & Okumoto, K. (1987). *Software reliability: measurement, prediction, application*. New York, NY, USA: McGraw-Hill, Inc.

Myers, G.J., Sandler, C., & Badgett, T. (2011). *The art of software testing*. John Wiley & Sons.

Nassar, D.M., Rabie, W.A., Shereshevsky, M., Gradetsky, N., Ammar, H.H., Yu, B., Bogazzi, S., & Mili, A. (2002). Estimating error propagation probabilities in software architectures[1]. *College of Computer Science, New Jersey Institute of Technology*.

National Security Agency. (2003). *Requirement specification for personnel access control system*.

Nautiyal, L., Gupta, N., & Dimri, S.C. (2014). Measurement of the reliability of a component-based development using a path-based approach. *ACM SIGSOFT Software Engineering Notes*, *39*(6), 1-5.

Nautiyal, L., Gupta, N., & Dimri, S.C. (2013). A new path based reliability approach for estimation of reliability of component based software development. *International Journal of Computer Science Engineering*, *2*(6), 295–299.

Pasquini, A., Crespo, A.N., & Matrella, P. (1996). Sensitivity of reliability-growth models to operational profile errors vs. testing accuracy [software testing]. *IEEE Transactions on Reliability*, *45*(4), 531-540.

Pham, H. (2006). *System software reliability*. Berlin: Spriner-Verlag. Online ISBN 978-1-84628-295-9.

Pham, T.T., & Defago, X. (2013, September). Reliability prediction for component-based software systems with architectural-level fault tolerance mechanisms. In *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on* (pp. 11-20). IEEE.

Popic, P., Desovski, D., Abdelmoez, W., & Cukic, B. (2005, November). Error propagation in the reliability analysis of component based systems. In *Software reliability engineering, 2005. ISSRE 2005. 16th IEEE international symposium on* (pp. 10-pp). IEEE.

Reussner, R.H., Schmidt, H.W., & Poernomo, I.H. (2003). Reliability prediction for component-based software architectures. *Journal of Systems and Software*, *66*(3), 241–252.

Rodrigues, G., Rosenblum, D., & Uchitel, S. (2005). Using scenarios to predict the reliability of concurrent component-based software systems. In M. Cerioli (Ed.), *Fundamental Approaches to Software Engineering* (pp. 111–126). Springer Berlin Heidelberg.

Sato, N., & Trivedi, K.S. (2007, July). Accurate and efficient stochastic reliability analysis of composite services using their compact Markov reward model representations. In *Services Computing, 2007. SCC 2007. IEEE International Conference on* (pp. 114-121). IEEE.

Schneidewind, N. (2009). Integrating testing with reliability. *Software Testing, Verification and Reliability*, *19*(3), 175–198.

Schneidewind, N.F., & Keller, T.W. (1992). Applying reliability models to the space shuttle. *IEEE Software*, *9*(4), 28–33.

Sharma, V.S., & Trivedi, K.S. (2007). Quantifying software performance, reliability and security: An architecture-based approach. *Journal of Systems and Software*, *80*(4), 493–509.

Shooman, M.L. (1973). Operational testing and software reliability estimation during program development. In I*EEE Symposium Computer Software Reliability*, pp. 51-57.

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*, Hong Kong, China, 2001, pp. 12-21). IEEE.

Sitaraman, M. & Weide, B.W. (1994). Special feature component-based software using RESOLVE. *ACM SIGSOFT Software Engineering Notes*, *19*(5), 21–67.

Tian, J., Rudraraju, S., & Li, Z. (2004). Evaluating web software reliability based on workload and failure data extracted from server logs. *IEEE Transactions on Software Engineering*, *30*(11), 754-769.

Voas, J. (1997). Error propagation analysis for COTS systems. *Computing Control Engineering Journal*, *8*(6), 269–272.

Wang, W.L., Wu, Y., & Chen, M.H. (1999). An architecture-based software reliability model. In *Dependable Computing, 1999. Proceedings. 1999 Pacific Rim International Symposium on* (pp. 143-150). IEEE.

Xie, M. (1991). *Software reliability modelling*. Vol. 1, World Scientific, Singapore.

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.

Zheng, Z., & Lyu, M.R. (2010, May). Collaborative reliability prediction of service-oriented systems. In *Software Engineering, 2010 ACM/IEEE 32nd International Conference on* (Vol. 1, pp. 35-44). IEEE.