IEC
Institute of Electronics and Computer

Journal of the Institute of Electronics and Computer, 2020, 2, 57-71
https://iecscience.org/journals/JIEC
ISSN Online: 2643-8240

Evaluation of Reliability in Component-Based System Using Architecture Topology

Fathollah Bistouni[1], Mohsen Jahanshahi[2,*]

[1] Department of Computer Engineering, Central Tehran Branch, Islamic Azad University, Tehran, Iran.
Email: fat.bistouni.eng@iauctb.ac.ir
[2] Department of Computer Engineering, Central Tehran Branch, Islamic Azad University, Tehran, Iran.
IEEE Senior Member, Email: mjahanshahi@iauctb.ac.ir
*Corresponding author

How to cite this paper: Fathollah Bistouni, Mohsen Jahanshahi (2020). Evaluation of Reliability in Component-Based System Using Architecture Topology. Journal of the Institute of Electronics and Computer, 2, 57-71.
https://doi.org/10.33969/JIEC.2020.21005.

Received: March 13, 2020
Accepted: March 18, 2020
Published: March 19, 2020

Abstract

In the component-based software, as its name indicates, the overall system performance is a reflection of the performance of its components. The correct analysis of reliability, which is known as a critical factor in component-based software engineering process, is one of the necessary tasks in such a system. However, most of the previous studies do not provide a practical and complete approach on this issue of this field. So, the aim of this work is to introduce a new systematic approach for software reliability analysis. The system architecture is used by this approach for time-dependent reliability evaluation.

Keywords

Component-based system, Reliability, System architecture, Mean time to failure

1. Introduction

Nowadays, the major role of software on our lives is not negligible, such that it can affect critical areas of our lives. Accordingly, reliability as one of the important aspects of quality of system components and proper connections among those are of great importance and it can indicate the critical position of software reliability engineering [1]. The probability of failure-free software operation or getting its expected precision is called software reliability [2]. Concerning this, the vital role of the application architecture in its performance and reliability is noticeable. Software architecture refers to “The visible structure of the system which comprises system components and the relations among them” [3]. Therefore, the component

reliabilities and the application architecture are the basement of the estimation of the application reliability which known as architecture-based system reliability analysis. So, one of its merits is to identify system weaknesses regarding reliability [4]. On the other hand, it can improve various performance metrics such as reliability, mean time to failure, and cost by architectural optimization. In this case, once the weaknesses of a software system are determined, reliability can be increased by directly performing of fault tolerance operations on the effective components.

Here, we reviewed the most important works in the field of architecture-based system reliability analysis, which have been performed as the result of the mentioned advantages. The goal of [1] was to provide an architecture-based reliability model which can consider heterogeneity of software architecture to address various component interactions. Accordingly, the method used in this work is based on discrete-time Markov chains as the building blocks for modeling application and calculating its reliability. However, some of major drawbacks of this work are as following: (1) This methodology was time-independent and time parameter didn't apply in this reliability analysis. (2) This type of analysis cannot fulfill the need of considering a large number of states and conditions in complex structures so it is useful just for analyzing simple architecture application. The other works in [4] resulted in giving an overview of the researches in the architecture-based system reliability analysis field, and its assumptions and limitations examination. In [5, 6], the purpose of author was to predict the performance and reliability of software systems according to their architecture by a hierarchical model. However, the prerequisite information of the techniques described in [5] are both the duration of visit in each component and each component reliability, as the fundamental assumption. Nevertheless, since more often than not, other sources are needed to access the time spent in each component, this method is difficult to apply practically. In addition, these hierarchical models in [5, 6] have an estimation approach compared to the composite models. Accordingly, composite models provide more accurate reliability metrics than these models. So, these solutions are not applicable in cases which critically require accuracy of the reliability prediction obtained from the architecture-based analysis. The aim of [6] was to introduce suitable calculation methods for the reliability and availability. Also, the effort of this work was the detection of defects of these methods. According to the analysis, it is not possible to supply all needs by none of the current methods. Component probability transition diagram as a general model, which is presented in Ref. [7], can support various types of components. Additionally, component-based system process is able to predict the reliability.

However, in the reliability analysis which is used in this work, the complex relationships between the components of the application were not included. Component-based system often has a complex architecture regarding reliability, so accurate reliability calculation is not possible through the approach introduced in this work. The new approach, which was discussed in [8], is a function of usage profiles and the reliability of required components while its aim is reliability analysis of the component-based application architectures. However, some demerits of this work are as follows: (1) Reliability analysis in this work is time independent and time parameter has no role in the analysis of reliability. (2) Derivation of the reliability equations which refers to a mathematical expression of the system reliability as a function of the reliability of system components does not take place in approach used in this work. The critical importance of these equations in the accurate reliability analysis as well as development of other reliability metrics such as mean time to failure (MTTF) and system failure rate is noticeable. In Refs. [9, 10], the aim is using path-based architectural reliability model to provide new approaches to analysis the reliability of component-based software system. However, some defects of these approaches are: (1) Path-based approaches, when presence of loops leads to application architecture with infinite paths, can't provide accurate application reliability. So, it is not suitable for complex architectural structures analysis accurately. (2) Derivation of the reliability equations components does not take place in approach used in this work (3) These analyses were time independent, which means that the time parameter is not included in the reliability analysis.

According to the discussion above, this paper has purposed to introduce a new approach to analyze reliability of component-based system using application architecture which will be discussed in section 2. Features of this approach are as following: (i) In this approach, reliability analysis is done while the relationships between the components were considered precisely. (ii) This approach is able to derive reliability equations. (iii) It is time dependent approach. So, the reliability equations are as a function of time. (iv) Both simple and complex architecture systems are suitable to be analyzed by the methodology used in this approach.

This paper will be continued as below: Section 2 will present the proposed approach. Section 3 will carry out reliability analyses. At last, some conclusions will be discussed in section 4.

2. The proposed approach

In this section, the proposed approach for the analysis of component-based system reliability is presented. This approach uses application architecture as connection

topology of the system components for reliability analysis. Also, it has some systematic steps, which will be explained step by step. First, it is useful that we explain the assumptions used in the proposed reliability analysis approach. These assumptions are as follows:

(1) Software components and their connection pattern are reflected by the application architecture. The success of component-based system depends on the ability of establishing required connections between system components. Therefore, the probability of failure-free connection between each given component (node) and another one (node) in the system architecture (topology/graph in graph theory) is known as reliability.

(2) One of the assumptions will be the likelihood of the failure of each connection (ink in graph theory) between the components in application architecture.

(3) All failures are statistically independent.

(4) The distribution of the failures is assumed to be exponential. Therefore, if λ be as the failure rate of a link, then the corresponding reliability is represented by: $R_L(t) = e^{-\lambda t}$.

(5) The architecture links have either working or failing state.

The steps of proposed approach to determine the reliability of the component-based system are as follows:

Step 1: Extraction of application architecture:

In this regard, an approach is based on clustering system components. Accordingly, the application components are grouping based on the degree of dependency i.e. components with the highest placed in a cluster. So, by clustering, the system architecture is more understandable and also the future maintenance operations are easier. Moreover, some clustering tools are used to extract system architecture. For instance, DAGC [11] is one of the tools which is helpful for this paper to reach its purpose.

Step 2: Calculation of reliability for each software cluster:

At this step, reliability is calculated for each cluster. Since the structure of clusters is often complex in terms of reliability engineering, we will use the decomposition method [12-16] to calculate the reliability of each cluster. In this method, which is an application of the law of total probability, the first step is selection of a key component that can be a link or cluster. Then calculation of the system reliability is done in two modes: activated key component and failed key component. Finally, the combination of these two probabilities is used to obtain the total system reliability. Therefore, based on the decomposition method, the reliability of a cluster can be calculated by Eq. (1).

$$R_{cluster}(t) = \overline{R_{key}}(R_{cluster} \mid \overline{R_{key}}) + R_{key}(R_{cluster} \mid R_{key}) \quad (1)$$

Step 3: Calculation of the cluster failure rate:

This step is a prerequisite for the main step namely calculation of reliability for the entire system. In this step, the objective is to obtain a failure distribution of the entire cluster based on the failure distribution of its software components. The parameter that can be used to study these cases is cluster failure rate. In fact, failure rate of a cluster is an indication of the proneness to failure of the cluster after time t has elapsed. The cluster failure rate, denoted λ_cluster, can be computed by the following equation [12, 16-18]:

$$\lambda_{cluster} = -\left(\frac{1}{R_{cluster}(t)}\right)\frac{d(R_{cluster}(t))}{dt} \quad (2)$$

Step 4: Calculation of reliability for the entire system:

In this step, the system reliability is calculated using the failure rate of clusters. For this purpose, it is assumed that each cluster is a node. Then, the connections between the clusters are considered to calculate the total reliability. For this purpose, we can use the following equation:

$$R_{software}(t) = \overline{R_{key}}(R_{software} \mid \overline{R_{key}}) + R_{key}(R_{software} \mid R_{key}) \quad (3)$$

Step 5: Calculation of mean time to failure (MTTF) for the entire system:

In the IT industry, “uptime” as an important reliability metric refers to the time that a system is available. For a system, the time span between outages or failures in which that system is online is known as the “time to failure”. The mean time to failure (MTTF) is defined as the average of the time to failure or the expected value of the time to failure. So, according to its crucial nature, this parameter is one of the most important performance metrics which will be investigated in this paper. The MTTF is calculated by [19, 20]:

$$MTTF_{software} = \int_0^{\infty} R_{software}(t)dt \quad (4)$$

In the next section, some case studies and the instance structures will be analyzed using the proposed approach. Therefore, in the next section, we will understand the proposed approach practically.

3. Case study

In this section, we examine a case study for a better understanding of the proposed approach. First, we consider the well-known Travelling Salesman Problem (TSP). The first step for applying the intended approach for reliability analysis is extraction

of TSP application architecture from its program. So, the DAGC tool is used to reach this goal. In DAGC tool, call graph and application architecture are its input and output, respectively. So, the first thing is TSP call graph extraction from its source code. Here, NDepend tool used to extract the call graph [21] which is then can be used as input for the DAGC in order to extract the application architecture. Fig. 1 represents the TSP software architecture extracted from its source code.

In step 2, the reliability should be calculated for each cluster. Thus, according to Eq. (1), reliability for the three clusters shown in Fig. 1 is given by:

$$R_{cluster\,1}(t) = e^{-2\lambda t} \tag{5}$$

$$R_{cluster\,2}(t) = e^{-3\lambda t} \tag{6}$$

$$\begin{aligned} R_{cluster\,3}(t) &= \left(1-e^{-\lambda t}\right)\left(\left(1-e^{-\lambda t}\right)\left(e^{-\lambda t}\left(5e^{-4\lambda t}-4e^{-5\lambda t}\right)\right)\right. \\ &+ e^{-\lambda t}\left(\left(1-e^{-\lambda t}\right)\left(5e^{-4\lambda t}-4e^{-5\lambda t}\right)+e^{-\lambda t}\left(2e^{-\lambda t}-e^{-2\lambda t}\right)\left(3e^{-2\lambda t}-2e^{-3\lambda t}\right)\right)\Big) \\ &+ e^{-\lambda t}\left(\left(1-e^{-\lambda t}\right)\left(2e^{-\lambda t}-e^{-2\lambda t}\right)\left(4e^{-3\lambda t}-3e^{-4\lambda t}\right)+e^{-\lambda t}\left(2e^{-\lambda t}-e^{-2\lambda t}\right)^{3}\right) \\ &= \frac{32}{e^{5t\lambda}}+\frac{54}{e^{7t\lambda}}-\frac{14}{e^{8t\lambda}}-\frac{71}{e^{6t\lambda}} \end{aligned} \tag{7}$$

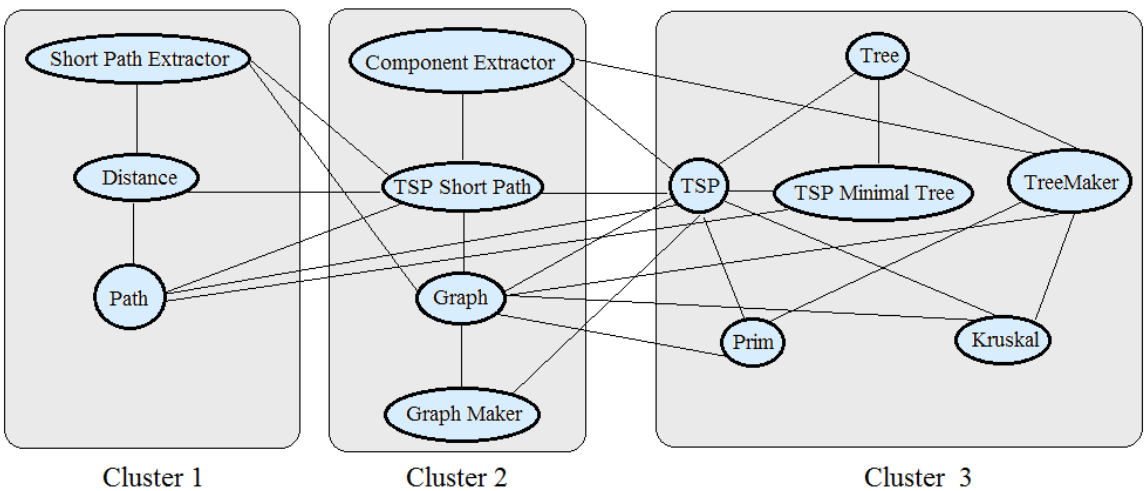

Figure 1. Electromagnetic spectrum

Now, according to Eq. (2), the failure rate of each cluster is calculated as follows:

$$\lambda_{cluster\,1} = -\left(\frac{1}{e^{-2\lambda t}}\right)\frac{d\left(e^{-2\lambda t}\right)}{dt} = 2\lambda \tag{8}$$

$$\lambda_{cluster\ 2} = -\left(\frac{1}{e^{-3\lambda t}}\right)\frac{d\left(e^{-3\lambda t}\right)}{dt} = 3\lambda \quad (9)$$

$$\lambda_{cluster\ 3} = -\left(\frac{1}{\frac{32}{e^{5t\lambda}}+\frac{54}{e^{7t\lambda}}-\frac{14}{e^{8t\lambda}}-\frac{71}{e^{6t\lambda}}}\right)\frac{d\left(\frac{32}{e^{5t\lambda}}+\frac{54}{e^{7t\lambda}}-\frac{14}{e^{8t\lambda}}-\frac{71}{e^{6t\lambda}}\right)}{dt} = \frac{\frac{160\lambda}{e^{5t\lambda}}+\frac{378\lambda}{e^{7t\lambda}}-\frac{112\lambda}{e^{8t\lambda}}-\frac{426\lambda}{e^{6t\lambda}}}{\frac{32}{e^{5t\lambda}}+\frac{54}{e^{7t\lambda}}-\frac{14}{e^{8t\lambda}}-\frac{71}{e^{6t\lambda}}} \quad (10)$$

At this point, based on Step 4, the reliability is calculated for the entire system. For this purpose, it is assumed that each cluster is a node. Then, the connections between the clusters are considered to calculate the total reliability considering the failure rate of clusters. Therefore, based on this change, Fig. 1 can be considered as Fig. 2.

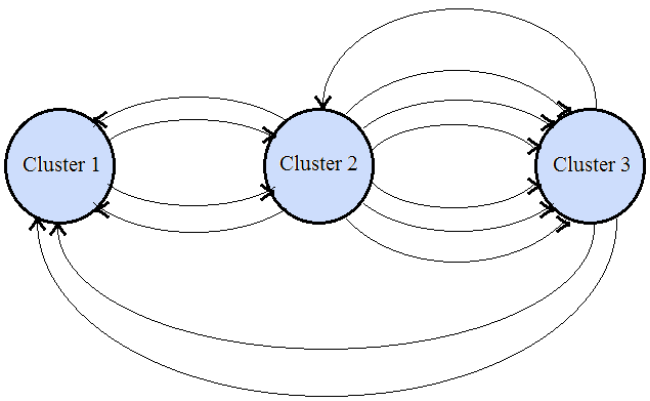

Figure 2. TSP software architecture by taking clusters as nodes.

To simplify of Fig. 2, it can be considered as Fig. 3, where the failure rates for each of the links 1, 2, and 3 are given by:

$$\lambda_{link\ 1} = -\left(\frac{1}{1-(1-e^{-\lambda t})^4}\right)\frac{d\left(1-\left(1-e^{-\lambda t}\right)^4\right)}{dt} = \frac{4\lambda\left(-\frac{3}{e^{t\lambda}}+\frac{3}{e^{2t\lambda}}-\frac{1}{e^{3t\lambda}}+1\right)}{-\frac{6}{e^{t\lambda}}+\frac{4}{e^{2t\lambda}}-\frac{1}{e^{3t\lambda}}+4} \quad (11)$$

$$\lambda_{link\ 2} = -\left(\frac{1}{1-(1-e^{-\lambda t})^7}\right)\frac{d\left(1-\left(1-e^{-\lambda t}\right)^7\right)}{dt} = \frac{7\lambda\left(-\frac{6}{e^{t\lambda}}+\frac{1}{e^{6t\lambda}}+\frac{15}{e^{2t\lambda}}+\frac{15}{e^{4t\lambda}}-\frac{6}{e^{5t\lambda}}-\frac{20}{e^{3t\lambda}}+1\right)}{-\frac{21}{e^{t\lambda}}+\frac{1}{e^{6t\lambda}}+\frac{21}{e^{4t\lambda}}+\frac{35}{e^{2t\lambda}}-\frac{7}{e^{5t\lambda}}-\frac{35}{e^{3t\lambda}}+7} \quad (12)$$

$$\lambda_{link\ 3} = -\left(\frac{1}{1-(1-e^{-\lambda t})^2}\right)\frac{d\left(1-\left(1-e^{-\lambda t}\right)^2\right)}{dt} = \frac{2\lambda\left(-\frac{1}{e^{t\lambda}}+1\right)}{-\frac{1}{e^{t\lambda}}+2} \quad (13)$$

Now, according to Fig. 3, it is clear that, (1) failure in any of the clusters will lead to total system failure; (2) the reliability of this architecture can be calculated by Eq. (3). With regard to the first point, the clusters are in series to each other in terms of

reliability. Therefore, based on Eqs. (8) through (10), we have:

$$\left(R_{closter\,1}(t)\right)\left(R_{closter\,2}(t)\right)\left(R_{closter\,3}(t)\right)=\left(e^{-(2\lambda)t}\right)\left(e^{-(3\lambda)t}\right)\left(e^{-\left(\frac{\frac{160\lambda}{e^{5t\lambda}}+\frac{378\lambda}{e^{7t\lambda}}-\frac{112\lambda}{e^{8t\lambda}}-\frac{426\lambda}{e^{6t\lambda}}}{\frac{32}{e^{5t\lambda}}+\frac{54}{e^{7t\lambda}}-\frac{14}{e^{8t\lambda}}-\frac{71}{e^{6t\lambda}}}\right)t}\right) \quad (14)$$

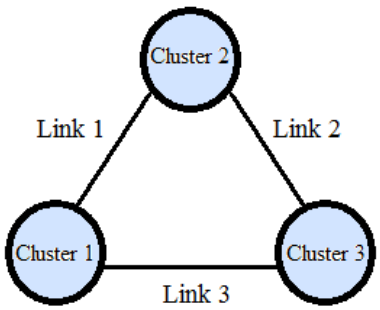

Figure 3. Simplification of TSP software architecture.

On the other hand, with regard to the second point, the reliability of the architecture in Fig. 3 can be calculated as follows (considering Eqs. (11) through (13)):

$$R_{Fig.3}(t)=\left(1-e^{-\left(\frac{4\lambda\left(-\frac{3}{e^{t\lambda}}+\frac{3}{e^{2t\lambda}}-\frac{1}{e^{3t\lambda}}+1\right)}{-\frac{6}{e^{t\lambda}}+\frac{4}{e^{2t\lambda}}-\frac{1}{e^{3t\lambda}}+4}\right)t}\right)\left(e^{-\left(\frac{7\lambda\left(-\frac{6}{e^{t\lambda}}+\frac{1}{e^{6t\lambda}}+\frac{15}{e^{2t\lambda}}+\frac{15}{e^{4t\lambda}}-\frac{6}{e^{5t\lambda}}-\frac{20}{e^{3t\lambda}}+1\right)}{-\frac{21}{e^{t\lambda}}+\frac{1}{e^{6t\lambda}}+\frac{21}{e^{4t\lambda}}+\frac{35}{e^{2t\lambda}}-\frac{7}{e^{5t\lambda}}-\frac{35}{e^{3t\lambda}}+7}\right)t}\right)\left(e^{-\left(\frac{2\lambda\left(-\frac{1}{e^{t\lambda}}+1\right)}{-\frac{1}{e^{t\lambda}}+2}\right)t}\right)$$
$$+e^{-\left(\frac{4\lambda\left(-\frac{3}{e^{t\lambda}}+\frac{3}{e^{2t\lambda}}-\frac{1}{e^{3t\lambda}}+1\right)}{-\frac{6}{e^{t\lambda}}+\frac{4}{e^{2t\lambda}}-\frac{1}{e^{3t\lambda}}+4}\right)t}\left(1\right.$$
$$\left.-\left(1-e^{-\left(\frac{7\lambda\left(-\frac{6}{e^{t\lambda}}+\frac{1}{e^{6t\lambda}}+\frac{15}{e^{2t\lambda}}+\frac{15}{e^{4t\lambda}}-\frac{6}{e^{5t\lambda}}-\frac{20}{e^{3t\lambda}}+1\right)}{-\frac{21}{e^{t\lambda}}+\frac{1}{e^{6t\lambda}}+\frac{21}{e^{4t\lambda}}+\frac{35}{e^{2t\lambda}}-\frac{7}{e^{5t\lambda}}-\frac{35}{e^{3t\lambda}}+7}\right)t}\right)\left(1-e^{-\left(\frac{2\lambda\left(-\frac{1}{e^{t\lambda}}+1\right)}{-\frac{1}{e^{t\lambda}}+2}\right)t}\right)\right) \quad (15)$$

To obtain the total system reliability, it should be noted that the reliability obtained in Eqs. (14) and (15) are in series to each other in terms of reliability. Therefore, the software reliability is calculated as follows:

$$R_{TSP}(t)=\left(R_{Fig.3}(t)\right)\left(\left(e^{-(2\lambda)t}\right)\left(e^{-(3\lambda)t}\right)\left(e^{-\left(\frac{\frac{160\lambda}{e^{5t\lambda}}+\frac{378\lambda}{e^{7t\lambda}}-\frac{112\lambda}{e^{8t\lambda}}-\frac{426\lambda}{e^{6t\lambda}}}{\frac{32}{e^{5t\lambda}}+\frac{54}{e^{7t\lambda}}-\frac{14}{e^{8t\lambda}}-\frac{71}{e^{6t\lambda}}}\right)t}\right)\right) \quad (16)$$

In the final step, the system MTTF can be calculated using Eq. (4):

$$MTTF_{TSP} = \int_0^{\infty} R_{TSP}(t)dt \quad (17)$$

According to Eq. (16), results of reliability analysis for the TSP as a function of time for different link failure rates are shown in Fig. 4. As the figure shows, by increasing the operating time, the reliability of the system considerably reduces, especially in higher link failure rates. In other words, application with poor connections between components is very prone to failure, especially for a long time. Therefore, one of the actions to improve the reliability of component-based system can be focused on reducing the connections failure rate. For this purpose, one method is to improve connection quality as well as fault tolerance by creating redundancy in components/links [22-24].

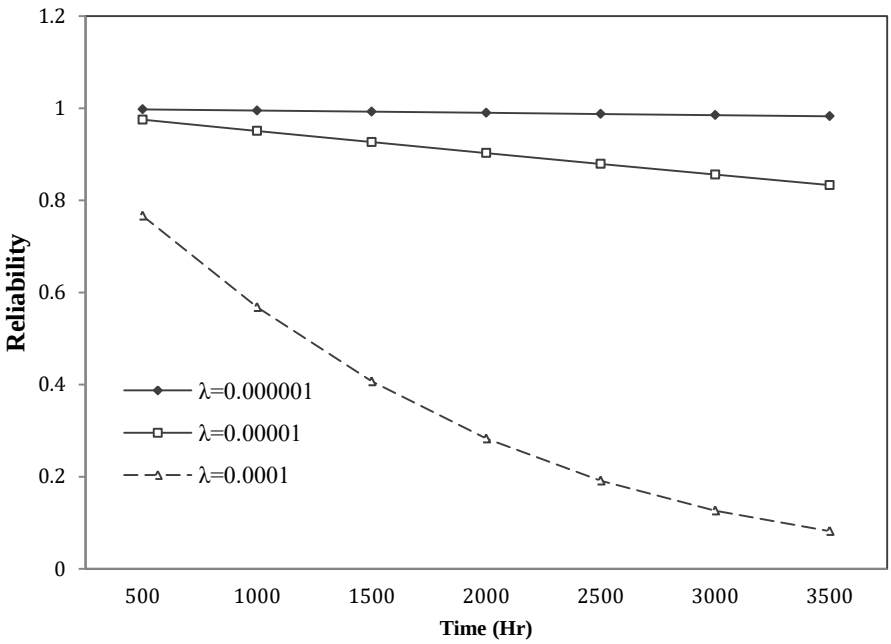

Figure 4. Reliability as a function of time.

In addition, in order to analyze the reliability of the system as a function of the reliability of the link, Fig. 5 can be considered. In this figure, the application reliability is shown for different link reliabilities. Fig. 5 shows that for low link reliability (0.4 and 0.5), TSP software reliability is very low and close to zero. This could be because there are many links in series in the TSP architecture topology, especially in clusters 1 and 2. On the other hand, increasing number of links in

series will increase the probability of failure and reduces the software reliability. Also, as it is showed in the figure, the more increasing the link reliability, the more significant improvement of the reliability of the entire system happens. Accordingly, improvement of the reliability of the system can be achieved by using some of redundant links in parallel or increasing link reliability.

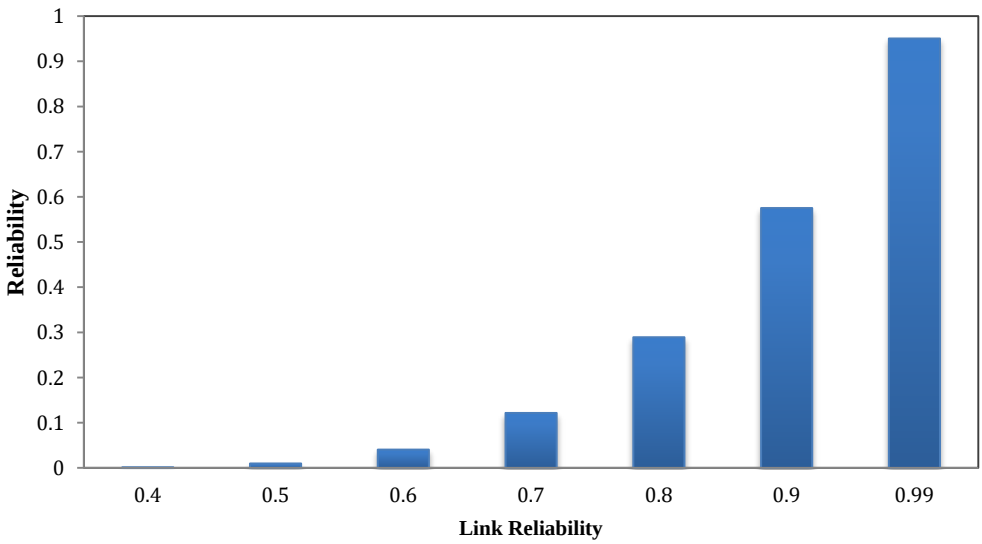

Figure 5. Reliability vs. link reliability.

Moreover, according to Eq. (17), results of MTTF analysis as a function of the link failure rate are shown in Fig. 6. As the figure shows, the increase in failure rate of connections between components has a major impact on reducing the MTTF of its system. In other words, it can be concluded that the improvement in the failure rate of connections between application components can lead to an increase in expected value of the time to software failure.

The last part of this section will be devoted to comparing our reliability analysis with the previous works. This comparison helps to provide the proof of the accuracy of the analyses used in this work, and extracts the advantages of our reliability analysis approach compared to other methods. To do this, we use one of the most accurate methods for determining the reliability of complex systems known as minimal path set-based method. Here, it is used to obtain the reliability of TSP. In a software architecture topology, a minimal path refers to a sequence of nodes and edges between source and destination while there are no cycles. Accordingly, the reliability of a system can be defined as the probability of the union of its minimal paths. In this regard, the efficient methodology which is used

in this work is enumeration of minimal paths sets before evaluating the reliability expression in compact form of sum of disjoint products (SDP) using improved multi-variable inversion (MVI) algorithm [25]. The time-independent reliability determination of the network topology systems can be done through this methodology [26, 27]. These algorithms are programming by means of MATLAB 9.0 platform. To consider the same reliability for all application components is the assumption of this method. In this method what we do are as following:

- To provide path sets for a particular pair of nodes.
- To generate desired order of the path sets based on cardinality.
- To achieve the path sets in ascending order of their cardinality.
- To evaluate terminal pair reliability from ordered minimal path sets using MVI algorithm.

Fig. 7 shows the results of time-independent reliability for TSP system, in which software reliability is represented for all possible values for the link reliability ($r \in [0, 1]$).

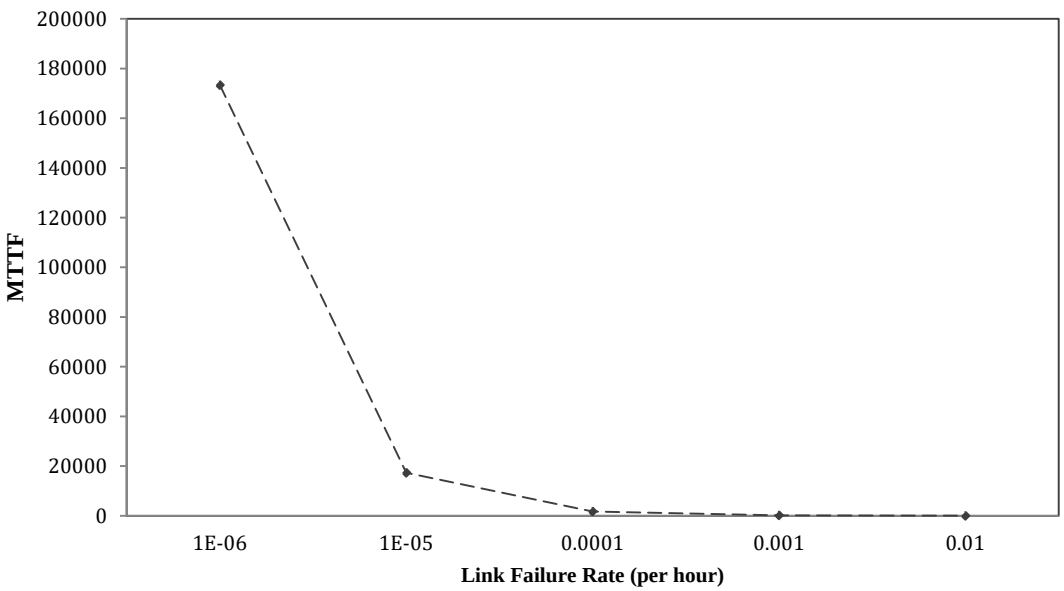

Figure 6. MTTF as a function of link failure rate.

Fig. 7 has a significant similarity with the results in Fig. 5. Fig. 7 shows that for low link reliability (from 0 to 0.5), TSP system reliability is very low and close to zero. In addition, as Fig. 7 shows, improving the reliability of the link could lead to a significant improvement in the reliability of the entire system.

These results represent the accuracy of the analysis method used in this work. In comparison with the analysis method used in this paper, minimal path set-based method has some defects despite its useful information providing, such as: (a) It is not suitable for complex system structures analysis. (b) Time is not among parameters which are considered in the method. (c) The reliability equations are difficult to Formulate.

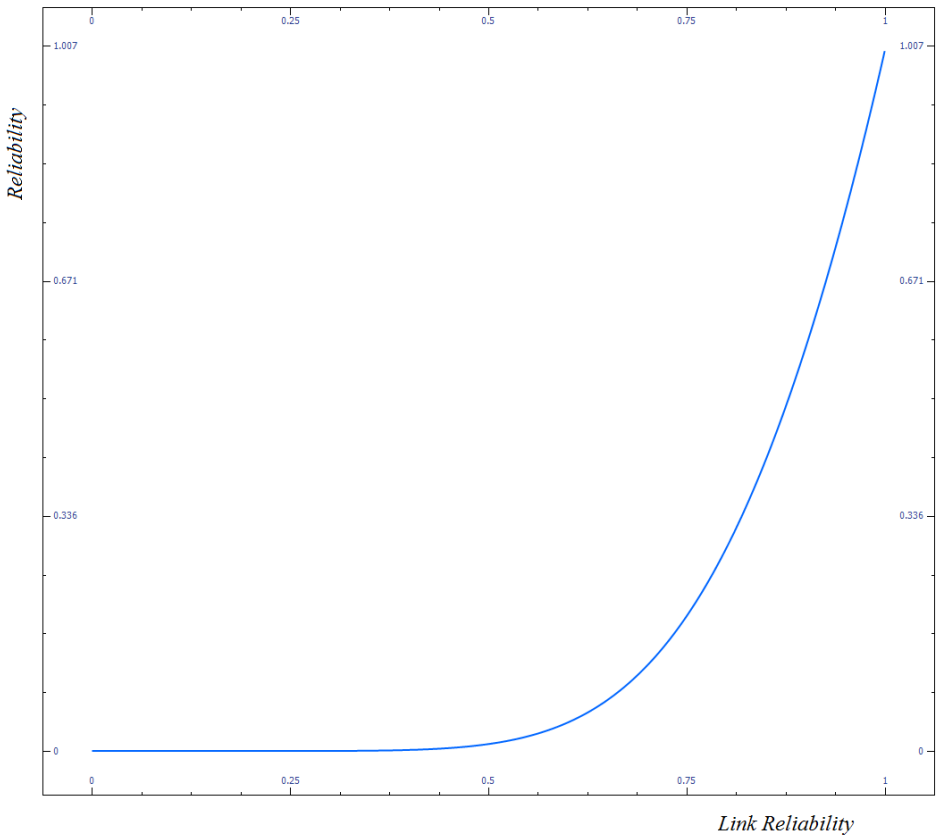

Figure 7. Reliability as a function of link reliability.

4. Conclusion and future works

This paper proposes a new systematic approach for component-based system reliability analysis. This approach has the following advantages over previous approaches: (1) In a detailed reliability analysis, it is able to consider the connections between application components carefully. (2) It computes the reliability equations. (3) In order to analyze the reliability, the operating time of application is considered. (4) The application components clustering is applied by this approach. Additionally, as a practical reliability analysis, a case study was

carried out. In which careful analysis of the reliability of Travelling Salesman Problem (TSP) is done through the proposed approach. Results of analyses showed that poor connections between components lead to a very unreliable system, particularly in a long operating time. Accordingly, creating redundancy in components/links to improve connection reliability along with fault tolerance can be a way to improve the reliability of component-based system. In addition, some studies can be considered as future works: One future research can be concentrated on examining the effects of different types of clustering on system reliability. Another idea is about finding optimal structures for application architecture in terms of reliability and fault tolerance.

Conflicts of Interest

There is no conflict of interest

References

[1] Luo, Yanan & Zou, Jie & Yao, Chengfei & Zhao, Xiaosong & Li, Tao & Bai, Gang. (2018). HSI-CNN: A Novel Convolution Neural Network for Hyperspectral Image. 464-469. 10.1109/ICALIP.2018.8455251.

[2] Wen-Li Wang, Dai Pan, and Mei-Hwa Chen. "Architecture-based software reliability modeling." Journal of Systems and Software 79.1 (2006): 132-146.

[3] Hai Hu, Chang-Hai Jiang, Kai-Yuan Cai, W. Eric Wong, and Aditya P. Mathur. "Enhancing software reliability estimates using modified adaptive testing." Information and Software Technology 55.2 (2013): 288-300.

[4] Tim Kelly. "Using software architecture techniques to support the modular certification of safety-critical systems." Proceedings of the eleventh Australian workshop on Safety critical systems and software-Volume 69. Australian Computer Society, Inc., 2007.

[5] Swapna S. Gokhale. "Architecture-based software reliability analysis: Overview and limitations." IEEE Transactions on dependable and secure computing 4.1 (2007): 32.

[6] Swapna S. Gokhale, and Kishor S. Trivedi. "Reliability prediction and sensitivity analysis based on software architecture." Software Reliability Engineering, 2002. ISSRE 2003. Proceedings. 13th International Symposium on. IEEE, 2002.

[7] Anne Immonen and Eila Niemelä. "Survey of reliability and availability prediction methods from the viewpoint of software architecture." Software & Systems Modeling 7.1 (2008): 49-65.

[8] Mao Xiaoguang and Deng Yongjin. "A general model for component-based software reliability." Euromicro Conference, 2003. Proceedings. 29th. IEEE, 2003.

[9] Ralf H. Reussner, Heinz W. Schmidt, and Iman H. Poernomo. "Reliability prediction for component-based software architectures." Journal of systems and software 66.3 (2003): 241-252.

[10] Fan Zhang, Xingshe Zhou, Junwen Chen, and Yunwei Dong. "A novel model for component-based software reliability analysis." In High Assurance Systems Engineering Symposium, 2008. HASE 2008. 11th IEEE, pp. 303-309. IEEE, 2008.

[11] Chao-Jung Hsu and Chin-Yu Huang. "An adaptive reliability analysis using path testing for complex component-based software systems." IEEE Transactions on Reliability 60.1 (2011): 158-170.

[12] Saeed Parsa and Omid Bushehrian. "A new encoding scheme and a framework to investigate genetic clustering algorithms." Journal of Research and Practice in Information Technology 37.1 (2005): 127.

[13] Israel Koren and C. Mani Krishna. Fault-tolerant systems. Morgan Kaufmann, 2010.

[14] Mohsen Jahanshahi and Fathollah Bistouni. Crossbar-Based Interconnection Networks: Blocking, Scalability, and Reliability. Springer, Switzerland, 2018.

[15] Fathollah Bistouni and Mohsen Jahanshahi. "Rearranging links: a cost-effective approach to improve the reliability of multistage interconnection networks." International Journal of Internet Technology and Secured Transactions 8.3 (2018): 336-373.

[16] Fathollah Bistouni and Mohsen Jahanshahi. "Reliability Analysis of Ethernet Ring Mesh Networks." IEEE Transactions on Reliability 66.4 (2017): 1238-1252.

[17] Fathollah Bistouni and Mohsen Jahanshahi. "Remove and contraction: A novel method for calculating the reliability of Ethernet ring mesh networks." Reliability Engineering & System Safety 167 (2017): 362-375.

[18] Fathollah Bistouni and Mohsen Jahanshahi. "Reliability analysis of multilayer multistage interconnection networks." Telecommunication Systems 62.3 (2016): 529-551.

[19] Fathollah Bistouni and Mohsen Jahanshahi. "Evaluating failure rate of fault-tolerant multistage interconnection networks using Weibull life distribution." Reliability Engineering & System Safety 144 (2015): 128-146.

[20] Mohsen Jahanshahi and Fathollah Bistouni. "A new approach to improve reliability of the multistage interconnection networks." Computers & electrical engineering 40.8 (2014): 348-374.

[21] Fathollah Bistouni and Mohsen Jahanshahi. "Formulating broadcast reliability equations on multilayer multistage interconnection networks." The Journal of Supercomputing 71.11 (2015): 4019-4041.

[22] Patrick Smacchia. "NDepend." Product description on company website at http://www.ndepend.com (2007).

[23] Laura L. Pullum. Software fault tolerance techniques and implementation. Artech House, 2001.

[24] Fathollah Bistouni and Mohsen Jahanshahi. "Analyzing the reliability of shuffle-exchange networks using reliability block diagrams." Reliability Engineering & System Safety 132 (2014): 97-106.

[25] Mohsen Jahanshahi and Fathollah Bistouni. "Improving the reliability of the Benes network for use in large-scale systems." Microelectronics Reliability

55.3 (2015): 679-695.

[26] S. K. Chaturvedi and K. B. Misra. "An efficient multi-variable inversion algorithm for reliability evaluation of complex systems using path sets." International Journal of Reliability, Quality and Safety Engineering 9.03 (2002): 237-259.

[27] S. Rajkumar, and Neeraj Kumar Goyal. "Reliability analysis of multistage interconnection networks." Quality and Reliability Engineering International 32.8 (2016): 3051-3065.

[28] S. Rajkumar, , and Neeraj Kumar Goyal. "Reliable multistage interconnection network design." Peer-to-Peer Networking and Applications 9.6 (2016): 979-990.