RESEARCH ARTICLE OPEN ACCESS

Comparative Analysis on Reliability Estimation Techniques of Component Based Software System

Gopal Prasad Jaiswal [1], Ram Nivas Giri [2], Dipesh Sharma [3]
Research Scholar [1], Assistant Professor [2], HOD [3]
Department of Computer Science and Engineering
RITEE, Raipur
Chhattisgarh - India

ABSTRACT
Software reliability means the probability of the uninterrupted operation of a software system for a specified period of time in a specified environment. Regularly software applications are swelling more complex and with more intensity on reuse. Then, Component Based Software (CBS) applications have appeared because CBS is an approach to software development that relies on software reuse. It turns up from the failure of object oriented development to backing equipped reusability. Single object classes are more extensive and unmistakable. Components are more abstract than object classes and can be considered to be stand- alone service providers. The focus of this paper is to give an analysis of Component Based Systems dependability estimation. In this paper, we will inspect diverse systems with respect to their degree, model, structures, system and support. This comparative analysis gives instinct into deciding the bearing of future CBS reliability research.

Keywords:- Reliability, Failure, Component Based Systems, Open Source Software, Reliability Model, Rule Based Model, Path Based Model, Additive Model, Operational Profile, Component Dependency Graph, Component, Failure Behavior, Flexibility, Understandability, Portability, Maintainability.

I. INTRODUCTION

Today's industry systems progressively depend on software which is constantly very complex. Software complexity increases also with the risk factor of the environment where the exclusive system is deployed. From these reasons the necessities for software reliability can't be overlook when outlining such framework [1]. Software reliability idea is one of industry's fundamental techniques for expecting the likelihood of software programming field failures. Software programming application reliability is defined as follows [1] [2]:

- "Reliability is a probability. This implies that failure happens arbitrarily, it can be a individual or repeating occasion. The frequency for failures fluctuates in spell of time as per the picked probability function".
- "The probability of a given framework works its assignment fairly for an altered time of time under the desired working conditions".
- "The probability of given program will safeguard perfection operation in a predefined domain for an altered spell of time".

Failure Probability is the probability that the software will fail on the back-to-back input selected. Software reliability is mostly measured per few unit of time, whereas probability of failure is broadly time autonomous. Software reliability varies vary from software "faultlessness". A project is faithful with its model, while reliability is related to the elementary request that is made upon the framework and the capacity to create a substantial reaction to those requests. A faultlessness program may be considered as unreliable, conversely a program that is not completely faultlessness may be considered as solid if the blunders are immaterial or client can just escape the errors.

Presently Component Based Software Engineering (CBSE) is being famous among both researchers and practitioners now that it's a lot of advantages over object oriented approach. CBSE has been an explicit result of advances in software component reuse. Outlining programming parts for future reuse has been a vital area in software engineering. Computing environments are evolving to distributed systems where Object-oriented development had not given wide reuse. Component-based software structure is a high level abstraction of system architecture. It has productive components: parts which give usefulness, connectors which portray intercommunication and setup which speaks to the topology of associations in the middle of segments. This abstraction gives a ton of force in the software product

life cycle like greater abstraction office, better adaptability for development and consideration, more reusability as relate to object oriented model. CBSE increase quality, productivity, profit and reusability and reduce care overheads and time to market [3] [4].

A mathematical fuzzy model based on necessity and possibility is proposed to predict the reliability of component-based software systems (CBSS). This paradigm doesn't require component failure data because it is based on uncertainty [5]. Software reliability estimation model based on Generic Algorithm (GA) and a Support Vector Machine (SVM). Software Reliability estimation factor for the SVM are determined by the GA. This model is less dependent on failure data than are other models [6]. An adaptive approach for testing path reliability estimation for complex CBSS. These use sequence, branch, and loop structures [7]. A rule-based approach based on fuzzy logic for estimating CBSS reliability. In this approach, four critical factors were recognized for estimating the reliability of a CBSS. They are utilized to plan a FIS for the estimation [8]. Another productive way to deal with access the reliability of a software project, considering the part reliabilities, relationship and application structural planning is arranged. That is transforms a Multivariate Burnoulli distribution (MVB) into a joint distribution of the component outcomes [9]. Software reliability estimation technique utilizing modified adaptive testing (MAT) for reliability of CBS. It can upgrade the software reliability estimation testing by controlling experiment test case selection process by providing more descriptive and exact results [10]. A versatile structure of incorporating path testing into reliability evaluation for component software systems. Three assessed strategies based on similar program namely, sequence, structures, loop structures and branch are conceded to figure the path reliability. Therefore, the borrowed path reliabilities can be adapted to the estimates of software reliability [11]. Conceptual model for reusability of the software is studied during improvement. The aspect of reusability of different versions are assessed and compared. The study helps to understand the evolution of software as per reusability. The change in size (ex. - aggregate number of lines of code, aggregate number of capacities, module and classes), flexibility, adaptability, reliability, complexity and understandability and so on are studied over in different releases to calculate their effect on reusability of the software program [12]. An adaptive neuro-fuzzy inference system (ANFIS) that is based on these two basic elements of soft computing and we compare its performance with that of a plain FIS (fuzzy inference system) for different data sets [13].

Rest of the paper is sorted out as follows Section 2 gives the problems mix with software reliability. Section 3 describes the reliability models for CBS. Section 4 gives the methodology for CBS reliability. In Section 5, result analysis of different approaches has been calculated. Paper is concluded with a summary and the description for future work in Section 6.

II. PROBLEMS DEFINATION WITH SOFTWARE RELIABILITY

The major difference in between software and other engineering affect is that software is pure design and maintaining. Software blunder means design error which is arising from human faults as well as machine error at the time of software development. Hardware systems do fail with respect to design and manufacturing faults. CBSS reliability is deeply relies on the intercommunication in between all the components. If number of intercommunication increases than dependencies also increases. Higher dependency increase the complexity of software application, Hence reliability calculation will be tedious work. Formally dependency is represented by an adjacency matrix. However, this representation can check only for the presence of dependencies in between components and does not consider the type of intercommunication. Intercommunication types have a significant contribution to the complexity of any CBS, hence many software reliability factors, issues and metrics proposed by researchers for CBS. Sharma et.al. [14] Component's have restricted intercommunication, data definition, protocols, flow chart paradigm [15]. We try to discover some factors for different types of dependency and intercommunication that requires special terminology. The factors that can be affected the dependence complexity of software system. That is [16].

- Data Dependence
- Control Dependence
- Interface Dependence
- Real Time Dependence

After this some other factors that is affected software reliability. They are: [13], [17]

(I) Reusability - Reusability (R) is a factor for estimating component reliability because components that have been used in many applications called more reliable.

Hence the reliability of a component is directly proportional to its reusability.

$$\text{Component Reliability} \propto \text{Reusability}$$

(II) Operational Profile - Operational profile (OP) is a complete set of operations with their probabilities of occurrence. The OP for any component describes the no. of inputs supplied to the component.

(III) Component Dependency - Various components or segments are interconnected to one or more component to form a larger application. These interconnected components are dependent on each other to execute their methods, because the output of one or more segments may use as an input for another component known as component dependency (CD). This dependency plays an important role for estimating the reliability of the whole program application. If segments are deeply dependent on one or more component, then reliability of the CBSS will be low.

(IV) Application Complexity - The application complexity (AC) of any CBSS application can be defined in terms of the number of components in that application and their interdependency. High complex and low reliable application means that more interdependent component. Hence, the application complexity (AC) of an application is inversely proportional to its reliability:

$$\text{Application Complexity} \propto (1 / \text{reliability})$$

(V) Flexibility – Software that can be easily changed as per different user and system requirements called flexibility.

(VI) Understandability - Understandability that means user can effectively understand whole application and working phenomena. After that users execute any task more effectively and efficiently. User leads to more efficiency because they can use functionality and all characteristics that achieve targeted goals faster, with little steps and errors.

(VII) Portability - Software application is considered portable if they have effortlessly ported to another environment without no exertion and adapt it to the new environment inside reasonable time limits.

(VIII) Maintainability – Computer program that can be easy to retain in its original form and to be restored to that form in case of a system failure called Maintainability.

III. RELIABILITY MODEL FOR COMPONENT BASED SYSTEMS

A software reliability models have emerged as people try to understand the feathers of how and why software untrustworthy, and try to estimate software reliability. Quantities of models have been proposed, yet how to use software reliability still dreary issue Software Reliability Models can be raised in two routes, one of the essential models in view of - [18]

(I) Failure History - With the assistance of failure history, the current Software reliability Model (SWRMs) can be ordered into four sections as:

- Time between Failure Models (TBF Models) - The methodology under study is the time between failures. It is expected that the time between (i-1)th and ith failures is an arbitrary variable, after a dispersion whose parameters rely on upon the quantity of faults staying in the system amid this interval.
- Fault Count Models (FC Models) - The irregular variable of hobby is the quantity of flaws (failures) happening amid determined time period. Failure finds out take after a known stochastic system. For the most part Poisson determination with a period ward will be discrete or ceaseless failure rate.
- Fault seeding Models (FS Models) - A Program has obscure number of indigenous issues. To this, a known number of faults are seeded. The project is tried and watched number of seeded and indigenous issue is numbered.
- Input domain based models (IDB Models) - An arrangement of experiments is produced from the info covering the operational profile (OP) of the program. Typically the information or data space is apportioned into a situated of comparable classes, each of which is generally connected with a project path.

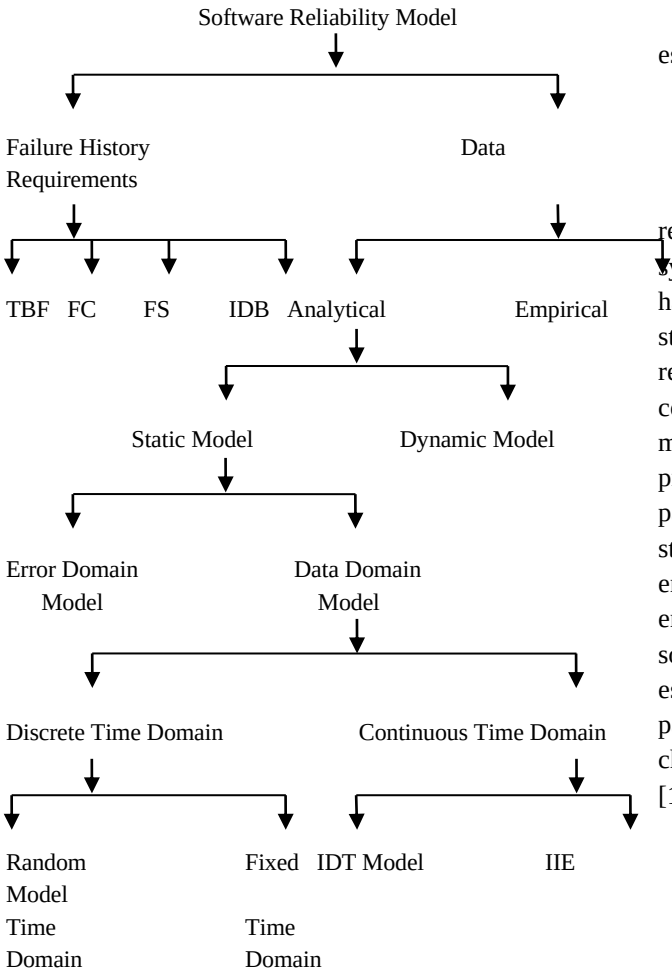

Fig. 1 Classification of software reliability model

(II) Data Requirements - On the premise of information necessities the SWRMs can be grouped into two fundamental groups-

- Empirical Models - An Empirical SWR model creates relationship or an arrangement of relationship between SWR measures and suitable programming measurements, for example, program complexity utilizing experimental results accessible from past information.
- Analytical Models - A diagnostic model requires some type of information accumulated from software failures. It is in view of fitting of a suitable dissemination with required assumptions for reliability on an arrangement of information assembled amid software testing and forecast of SWR parameters from the fitted conveyance.

IV. METHODOLOGIES

Researchers have proposed a number of methods for estimating CBSS reliability. They are following-

(1) Path based software reliability estimation

This is an adaptive approach for testing path into reliability estimation for complex component based systems. For path reliability estimation three methods have been proposed namely sequence, branch and loop structures. The proposed path reliability can be used for reliability estimation of overall application. [7]. Many computer-aided software engineering (CASE) tools or modeling languages [19], [20], [21], [22] have been provided to construct a graphic representation for a particular programming framework, for example, control stream diagram, component dependency graph (CFG), entity relationship diagram (ERD), and so on. With the end goal of describing the failure behavior of modular software systems, a path-based method is proposed to estimate the software reliability. The general structure of path reliability with crossing some specific nodes can be characterized by utilizing a various leveled methodology [19], [23], [24], [25]–[27].

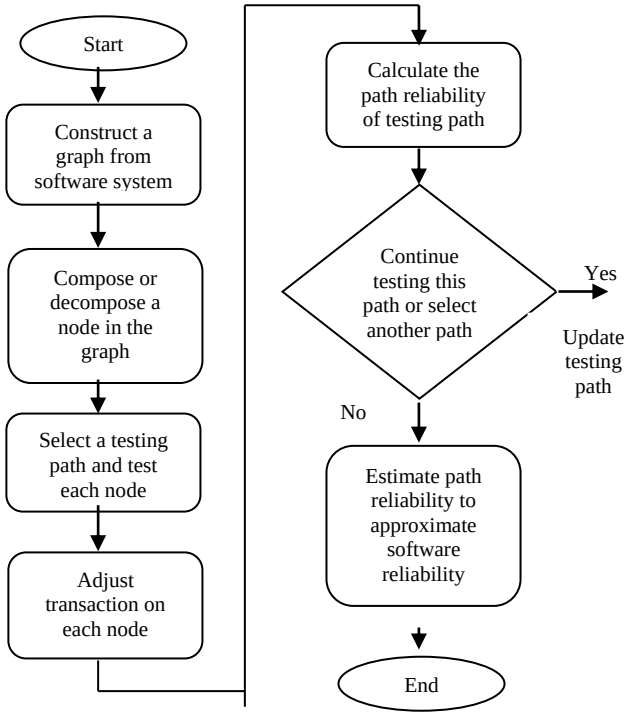

Fig. 2 Flow chart for path based software reliability estimation

(2) Architecture based software reliability estimation

In this approach, five fundamental segments or component composition mechanisms and their reliability evaluation strategies are given. After this evaluation the reliability for each composition, a methods and system is given to estimate the complete application reliability. Five segment composition mechanisms and their reliability formalizations are recognized in this section. They characterize (Ci , Ri , Ni) to speak the nodes in state chart, where Ci is the name of the ith segment or component, Ri is segment or component reliability, and Ni is the execution time period [28]. There utilized notations is-

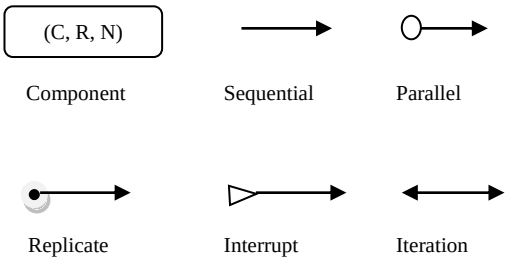

Fig. 3 Composition mechanism notations

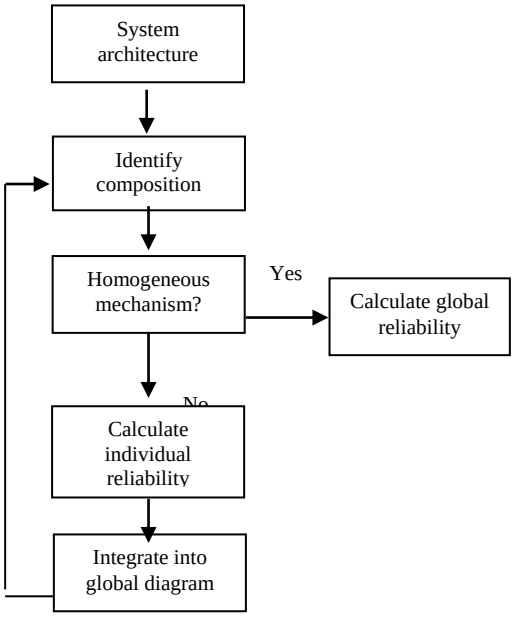

Fig. 4 Architecture-based reliability estimation framework

(3) Rule based software reliability estimation

The algorithm for rule based software reliability estimation consists of the following steps: [8]

1. Identify the factors that affect CBS reliability and methods.
2. Summarize a database for the value of these factors.
3. Apply best clustering technique and design clusters of these factors.
4. Design an inference engine for the rule base, with respect to planned clusters.
5. Fuzzify the inputs.
6. Defuzzify the outputs.
7. Estimate the error percentage.

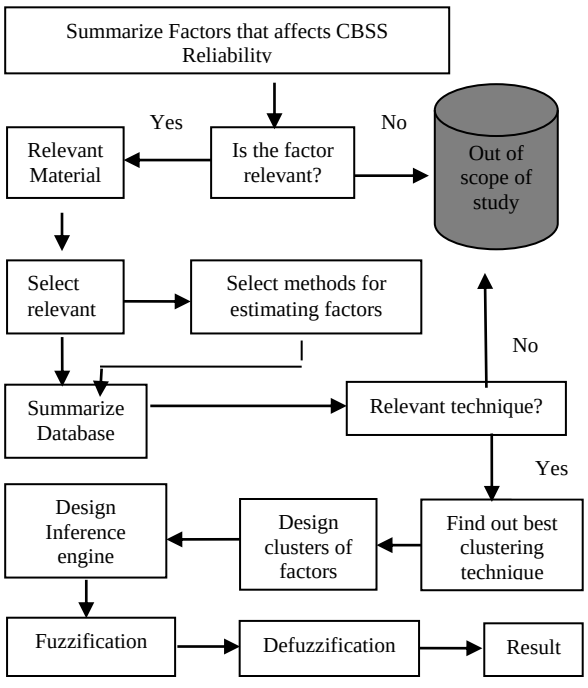

Fig. 5 Flow chart for rule based software reliability estimation

(4) Software reliability estimation based on Neuro-Fuzzy model

They are black-box models. NNs and fuzzy logic are reciprocal ideas. NNs have learning abilities: they can gain from information and criticism. Fuzzy logic models are rule-based models, where the rules usually have an if-then form. Fuzzy models do not have learning capabilities, so for learning, they must adopt techniques from other methods [13] [29].

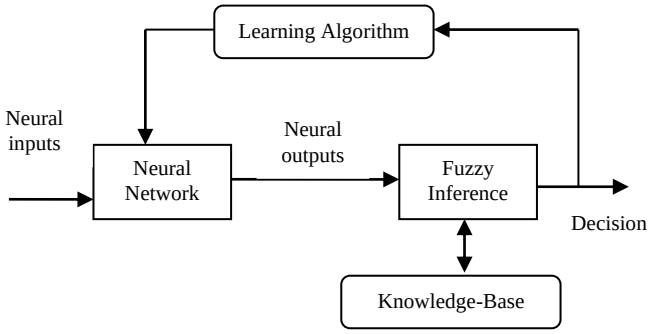

Fig. 6 Neural-Fuzzy system

V. RESULTS

In this paper, we have explained four methodologies. Software reliability of CBSS with the basis of Path based software reliability estimation 0.283, similarly Architecture based software reliability estimation 0.984, Rule based software reliability estimation 0.75 and Software reliability estimation based on Neuro-Fuzzy model 0.383.

VI. CONCLUTIONS AND FUTURE SCOPE

In this paper, various available reliability estimation methods for component based software applications are investigated. We study some regions on the premise of which we researched the accessible methodologies as extension, model, procedure, system strategy and study. Most of the proposed approaches are mathematical and based upon the component. To calculate the overall application reliability existing work take two important considerations one is reliability of individual component and another is Open Source Software (OSS) software product lines of the system. However, soft computing techniques have produced better results. For reliability there is still a good scope to estimate it using adaptive soft computing techniques.

REFERENCES

[1] Martin Jedlicka, Oliver Moravcik, Peter Schreiber, "Survey to Software Reliability" Pavlinska 16, 91724 Trnava, Slovakia.

[2] ANSI/IEEE, 1991 Standard Glossary of Software Engineering Terminology, STD-729-1991.

[3] Kirti Tyagi and Arun Sharma, 2012, "Reliability of Component Based Systems – A Critical Survey", Issue 2, Volume 11, E-ISSN: 2224-2872.

[4] NehaBudhija, Bhupinder Singh and Satinder Pal Ahuja, 2013, "Detection of Reusable Components in Object Oriented Programming Using Quality Metrics", Volume 3, Issue 1, ISSN: 2277 128X.

[5] Dimov, Aleksandar, Sasikumar, Punnekkat, 2010. "Fuzzy reliability model for component-based software systems", 36th EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 39–46.

[6] Lo, J., 2010. "Early software reliability prediction based on support vector machines with genetic algorithms", Fifth IEEE Conference on Industrial Electronics and Applications, pp. 2221–2226.

[7] M Hsu, C., Huang, C., 2011, "An adaptive reliability analysis using path testing for complex component based software systems", IEEE Trans. Reliab. 60 (1), 158–170.

[8] Tyagi, K., Sharma, A., 2012, "A rule-based approach for estimating the reliability of component-based systems", Adv. Eng. Softw. 54, 24–29.

[9] Fiondella, Lance, Rajasekaran, Sanguthever, Gokhale, Swapana, 2013. "Efficient software reliability analysis with correlated component failures". IEEE Trans. Reliab. 62 (1), 244–255.

[10] Hai Hu, Chang-Hai Jiang, Kai-Yuan Cai, W. Eric Wong, Aditya P. Mathur, 2013. "Enhancing software reliability estimates using modified adaptive testing", Information and Software Technology Journal Elsevier, pp. 288–300.

[11] Chao-Jung Hsu and Chin-Yu Huang, 2011, "An Adaptive Reliability Analysis Using Path Testing for Complex Component-Based Software Systems", IEEE TRANSACTIONS ON RELIABILITY, VOL. 60, NO. 1.

[12] Fazal-e-Amin, Ahmad Kamil Mahmood and Alan Oxley, 2012, "An Evolutionary Study of Reusability in Open Source Software", International Conference on Computer & Information Science (ICCIS) 978-1-4673-1938-6/12.

[13] Kirti Tyagi, Arun Sharma 2014, "An adaptive neuro fuzzy model for estimating the reliability of component-based software systems" Applied Computing and Informatics (2014) 10, 38–51.

[14] Arun Sharma, P.S.Grover, and Rajesh Kumar, 2009," Dependency Analysis for Component Based Software Systems", ACMSIGSOFT Software Engineering Notes Vol. 34, No 4 pp 1-6.

[15] Hapner et al., "Patterns of Conflict among Software Components." Volume 79, 2006, pp. 537-551.

[16] Ratneshwer and Anil Tripathi, 2011, "Dependence Analysis of Component Based Software through Assumptions", IJCSI, Vol. 8, Issue 4, No 1.

[17] Fazal-e-Amin, Ahmad Kamil Mahmood, Alan Oxley, 2012 "An Evolutionary Study of Reusability in Open Source Software", International Conference on Computer & Information Science (ICCIS).

[18] Latha Shanmugam and Dr. Lilly Florence, 2012, "An Overview of Software Reliability Models", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 10.

[19] S. S. Gokhale, 2007, "Architecture-based software reliability analysis: Overview and limitations," IEEE Trans. Dependable and Secure Computing, vol. 4, no. 1, pp. 32–40.

[20] K. Go seva-Popstojanova and K. S. Trivedi, 2001, "Architecture-based approach to reliability assessment of software systems," Performance Evaluation, vol. 45, no. 2/3, pp. 179–204.

[21] G. J. Myers, 2004, "The Art of Software Testing", 2nd ed. New York: John Wiley and Sons.

[22] P. C. Jorgensen, Software Testing: A Craftman's Approach, 3rd ed. : Auerbach Publications, 2008.

[23] S. S. Gokhale, W. E. Wong, J. R. Horgan, and K. S. Trivedi, 2004, "An analytical approach to architecture-based software performance and reliability prediction," Performance Evaluation, vol. 58, no. 4, pp. 391–412.

[24] Z. Zheng and M. R. Lyu, 2010. "An adaptive QoS-aware fault tolerance strategy for web services," Empirical Software Engineering, vol. 15, no. 4, pp. 323–345.

[25] J. H. Lo, S. Y. Kuo, M. R. Lyu, and C. Y. Huang, 2002, "Optimal resource allocation and reliability analysis for component-based software applications," in Proceedings of the 26th International Computer Software and Applications Conference (COMPSAC 2002), Oxford, England, pp. 7–12.

[26] J. H. Lo, C. Y. Huang, S. Y. Kuo, and M. R. Lyu, 2003, "Sensitivity analysis of software reliability for component-based software applications," in Proceedings of the 27th International Computer Software and Applications Conference (COMPSAC 2003), Dallas, TX, USA, pp. 500–505.

[27] J. H. Lo, C. Y. Huang, I. Y. Chen, S. Y. Kuo, and M. R. Lyu, 2005, "Reliability assessment and sensitivity analysis of software reliability growth modeling based on software module structure," Journal of Systems and Software, vol. 76, no. 1, pp. 3–13.

[28] Si Yuanjie, Xiaohu Yang, Xinyu Wang, Chao Huang, Aleksander J. Kavs, 2011. "An architecture-based reliability estimation framework through component composition mechanisms", 2nd International Conference on Computer, Engineering and Technology, pp. 165–170.

[29] Azar, Ahmad Taher, 2010, "Adaptive Neuro-Fuzzy Systems", ISBN 978-953-7619-92-3, pp. 216.