Global Journal of Computer Science and Technology
Software & Data Engineering
Volume 13 Issue 11 Version 1.0 Year 2013
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Performance Analysis of Improved Component Based Software Reliability Model

By K. Venkata Subba Reddy & Dr. B. Raveendra Babu
Muffakham Jah College of Engineering and Technology, India

Abstract - Software reliability engineering techniques focus on development and maintenance of software systems. This paper presents a improved component model. The model is used to estimate the reliability of software systems and the usage ration of each component. A component impact analysis which helps in focusing of testing is presented .The proposed method exhibits considerable improvement when compared against conventional methods.

Keywords : *software reliability, component ratio, reliability estimation model.*

GJCST-C Classification : *D.2.4*

Strictly as per the compliance and regulations of:

Performance Analysis of Improved Component Based Software Reliability Model

K. Venkata Subba Reddy [α] & Dr. B. Raveendra Babu [σ]

Abstract - **Software reliability engineering techniques focus on development and maintenance of software systems. This paper presents a improved component model. The model is used to estimate the reliability of software systems and the usage ration of each component. A component impact analysis which helps in focusing of testing is presented .The proposed method exhibits considerable improvement when compared against conventional methods.**

Keywords : software reliability, component ratio, reliability estimation model.

I. Introduction

There is probably no other human made material which is more Omni present than software in our modern technical worlds. New applications softwares are coming into the market day by day to meet the human requirements. Software has become one of the key parts of many aspects of society like banking, telecommunications, automobiles, aviation, shopping, auditing, web teaching, personal entertainment, and so on. Today, science and technology demand high quality software for making improvements and breakthroughs [1].

SRE is mainly focused on key properties like reliability that can be defined as probability of a failure free operation over a period of time under specified environment [2]. Among many other properties like functionality, flexibility, capability and so on reliability has got its own importance and accepted as the major factor in quality assessment of software. This assessment quantifies failures that make the powerful systems to inoperative. SRE is defined as the qualitative study of the operational behavior of software based systems with respect to user requirements concerning reliability. As a proven technique, SRE has become a standard and currently practiced by more than 50 organizations in their software projects and reports including AT&T, Lucent, IBM, NASA, Microsoft, and many others in Europe, Asia, and North America. However, this number is still relatively small compared to the large amount of software producers in the world.

Author α : Assistant Professor, Department of Computer Science and Engineering, Muffakham Jah College of Engineering and Technology, Hyderabad, Andhra Pradesh, India.
E-mail : kvsreddy2012@gmail.com
Author σ : Professor and Head, Department of Computer science and Engineering, VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad, Andhra Pradesh, India.
E-mail : rbhogapathi@yahoo.com

Reliability measurements are useful in supporting management of the software development process. For instance, obtaining reliability estimates early in the development process can help determine if the software system is on track to meet its reliability goals and therefore increase management effectiveness. Because research in software reliability is both necessary and beneficial, this paper proposes further work in this field of software engineering. In particular, efforts are focused towards structure based software reliability models.

This paper is organized as; section II discuss about the history of SR and the structural model, Section III explains about the proposed component model for SR estimations ending with section IV and V about the results and discussions.

II. Background

a) Software Structure Model

Software structure model can be defined by the components from which the software system is built and by establishing the relationships between these components. Many methods are available for the designing of software systems, and these methods provide a framework for decomposing the problem of software design into smaller pieces that are related to one another. Because the design process ultimately provides a blueprint for implementation, software design methods can be used to describe how software is physically decomposed. The two most popular categories of software design methods are Structured Design (SD) and Object-Oriented Design (OOD). Previous work on structural reliability models assumes that software is designed according to SD principles [3, 4, and 5]. The proposed model assumes that software is designed according to OOD principles.

Relationships are the basis of methods for calculating a total software system reliability figure from the reliability of the components. Current structural models follow one of two trends: execution path based [6, 7]. The practicality of the execution path approach is questionable; execution paths are preferred to the transition probability models because of the Markov process which does not reflect reality. In this paper, execution paths are used; however, execution paths are considered as series of component executions and reliability of a path will be obtained from the reliability of the path components.

Although the problems associated with execution paths can now potentially arise, the mathematical formulas derived for the software system reliability estimation will be simplified in a way that prior knowledge of the set of all execution paths is not necessary. Also, it is important to note that the combination of relationships and components that form the software structure model is extremely simplified. As software systems scale up, it might be necessary, for practicality purposes, to provide other components to group sub-components and other relationships.

b) Component Model

As stated in [7] shooman's model provides an equation to estimate the expected number of software failures (n_f)

$$n_f = Nf_1q_1 + Nf_2q_2 + \ldots\ldots\ldots\ldots + Nf_nq_n \quad (1)$$

Where n is the number of software tests, f_1 is the execution path usage ratio and q is the probability of failure of the path and n is the number of execution paths. Using the above equation (1) the software system probability of failure Q_s can be defined as

$$Q_s = \frac{n_f}{N} = \sum_{i=1}^{n_1} f_i q_i \quad (2)$$

The above equation describes the system unreliability is equal to the sum of the likelihood of failure over every execution path weighted by its corresponding execution path usage ratio. The software reliability R_s is defined as

$$R_s = 1 - Q_s \quad (3)$$

III. Methodology

The probability of a successful path represents that all components in the path execute successfully. In terms of conventional reliability theory, execution paths are equivalent to series reliability models. Series models require that all components work properly for system success. For independent component failures, the probability of failure along a given execution path is solely a function of the probability of failure of each component along the path.

$$p_i = p(C_{i1}).p(C_{i2}) \ldots\ldots P(C_{in}) \quad (4)$$

Where n_i is the number of components on the respective path

$$p_i = \prod_{j=1}^{n} C_{ij} = \prod_{j=1}^{n}(1 - d_{ij}) \quad (5)$$

Unreliability can be written as

$$q_i = 1 - \prod_{j=1}^{n}(1 - d_{ij}) \quad (6)$$

If the components have a high reliability then the above equations can re written as

$$p_i = 1 - \sum_{j=1}^{n} d_{ij} \quad (7)$$

$$q_i = \sum_{j=1}^{n} d_{ij} \quad (8)$$

Therefore software reliability can be defined as

$$R_s = 1 - Q_s = 1 - \sum_{k=1}^{m} \varphi_k D_k \quad (9)$$

Where D_k represents the probability of failure of component.

The component usage ratio parameter represents the ratio of total component execution time over the total software system execution time. The value of the component usage ratio is . The total of all components usage ratios is equal to 1. The component usage ratio weights the component reliability impact of the component reliabilities on the overall software system reliability. As a general the calculation of software reliability based rule, the reliability of a component frequently on component testing and/or inspection does not executed is expected to have more impact on the produce an accurate estimate of the reliability of overall system reliability than a component rarely the software system. When components are executed

$$\varphi_k = \frac{t_k}{T_s} \quad (10)$$

Calculation of component execution time is more difficult because it is necessary to track when each component is executed. The total component execution time describes the time spent executing instructions inside a component.

IV. Results and Discussions

Table 1 : Example Data

K	C_k	D_k	φ_k
1	0.96	0.04	0.2
2	0.96	0.04	0.02
3	0.96	0.04	0.001
4	0.98	0.02	0.05
5	0.98	0.02	0.05
6	0.98	0.02	0.05
7	0.99	0.01	0.05
8	0.99	0.01	0.02
9	0.99	0.01	0.006
10	0.99	0.01	0.006
11	0.99	0.01	0.2
12	0.99	0.001	0.07
13	0.99	0.001	0.02
14	0.99	0.001	0.001
15	0.99	0.001	0.001

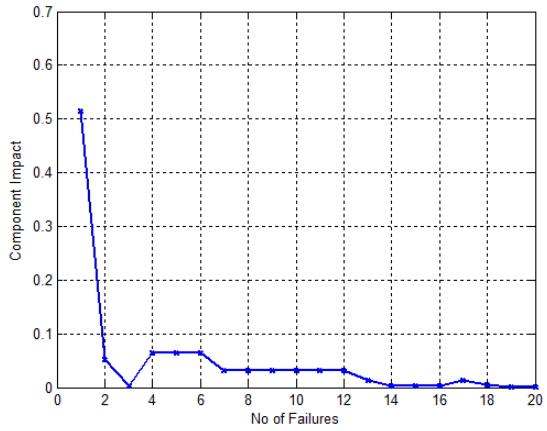

Figure 1 : Performance analysis of component Impact

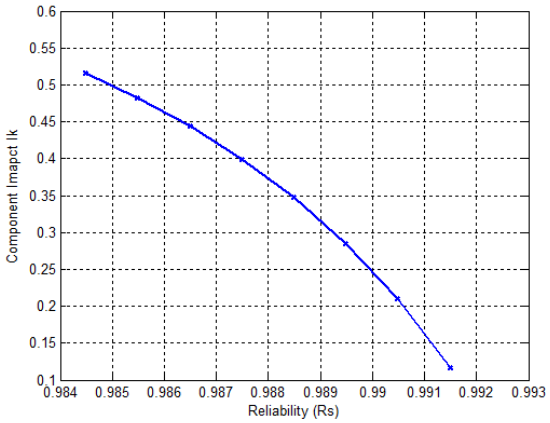

Figure 2 : performance analysis of component impact with respect to reliability

In this experiment we have considered a data as shown in the table 1. The above figure shows the impact of components with respect to the failures and the reliability. It is clear that the components reliability plays a prominent role in deciding the quality of the software. Numbers of failures are very low at a higher component impact and decreases gradually when it comes for the reliability.

V. Conclusions

A new mathematical model of software reliability is proposed in this paper. This focuses mainly on the impact of components when the reliability of software is estimated. It is proved that the malfunctioning of components have a great impact on the estimated reliability of software and the number failures are not optimal for calculations.

References Références Referencias

1. Michael R. Lyu, “software Reliability Engineering: A road map”, IEEE conf on .future of software engineering (FOSE-2007).
2. ANSI/IEEE, Standard Glossary of Software, Engineering Terminology, STD-729-1991, ANSI/IEEE, 1991.
3. P. Kubat, "Assessing Reliability of Modular Software", Operations Research Letters, Vol 8, No. 1, February 1989, pp. 35-41.
4. B. Littlewood, "Software Reliability Model for Modular Program Structure", IEEE Transactions on Reliability, Vol. R-28, No. 3, August 1979, pp. 241-246.
5. Y. Masuda, N. Miyawaki, U. Sumita and S. Yokoyama, "A Statistical Approach for Determining Release Time of Software System with Modular Structure", IEEE Transactions on Reliability, Vol. 38, No. 3, August 1989, pp. 365-372.
6. M.L. Shooman, "Software Engineering: Design, Reliability and Management", McGraw Hill 1983, ISBN 0-07-057021-3.
7. M.L. Shooman, "A Micro Software Reliability Model for Prediction and Test Apportionment", Proceedings 1991 International Symposium on Software Engineering (Austin, Texas), May 1991, pp. 52-59.
8. D.L. Parnas and Y. Wang, "The Trace Assertion of Module Interface Specification", Technical Report 89-261, Queen's University, October 1989.
9. R.C. Cheung, "A User-Oriented Software Reliability Model", IEEE Transactions on Software Engineering, Vol. SE-6, No. 2, March 1980, pp. 118-125.
10. G. Booch, "Object-Oriented Analysis and Design with Applications", The Benjamin/Cummings Publishing Company Inc.1994, ISBN 0-8053-5340-2.

This page is intentionally left blank