International Journal of Mechanical and Industrial Engineering

Volume 2 | Issue 4 Article 15

April 2013

Reliability Estimation Model for Software Components Using CEP

R. Chinnaiyan
A.V.C. College of Engineering , Mayiladuthurai, Tamil Nadu, R.Chinnaiyan@gmail.com

S. Somasundaram
Coimbatore Institute of Technology, Peelamedu- Coimbatore, S.Somasundaram@gmail.com

Reliability Estimation Model for Software Components Using CEP

R. Chinnaiyan[1] & S. Somasundaram[2]
[1]Deparment of CA, A.V.C. College of Engineering , Mayiladuthurai, Tamil Nadu – INDIA
[2]Department of Mathematics , Coimbatore Institute of Technology, Peelamedu- Coimbatore , Tamil Nadu - INDIA

Abstract - This paper presents a graphical complexity measure based approach with an illustration for estimating the reliability of software component. This paper also elucidates how the graph-theory concepts are applied in the field of software programming. The control graphs of several actual software components are described and the correlation between intuitive complexity and the graph-theoretic complexity are illustrated. Several properties of the graph theoretic complexity are presented which shows that the software component complexity depends only on the decision structure. A symbolic reliability model for component based software systems from the execution path of software components connected in series, parallel or mixed configuration network structure is presented with a crisp narration of the factors which influence computation of the overall reliability of component based software systems. In this paper, reliability estimation model for software components using Component Execution Paths (CEP) based on graph theory is elucidated.

Keywords *-Software Component, Complexity Measure, Component Execution Paths*

I. INTRODUCTION

IEEE 610.12-1990 defines reliability as “The ability of a software system or component to perform its required functions under stated environmental conditions for a specified period of operating time.” IEEE 982.1-1988 defines Software Reliability Management as “The process of optimizing the reliability of software through a program that emphasizes software error prevention, fault detection and removal, and the use of measurements to maximize reliability in the light of project constraints such as Resources, Schedule and Performance.” Evaluating the reliability of Component Based Software Systems is useful in quantifying the quality of the software systems. However, these quality measurements are now being implemented during development process leave too little to be done to improve the quality of the component based software system in an economic way. In the software design perspective, various methods for modeling software systems and specifying their functionality have been developed. These methods enable extensive analysis of the specification, but typically lack quantification.

II. RELATED WORK

P K Suri et al. [12] made an attempt to compute the reliability of the system as a function of reliabilities of its components. Components along a path, called Course-of-Execution, are executed during each simulation run. Starting from any component during any Course-of-Execution, control is transferred to other components as per the Markov process. Tirthankar Gayen et al.[19] presented a unique methodology based on the execution scenario analysis of the (COTS) component based software application to regain some control over their COTS component based software application systems by predicting the upper and lower bounds on the reliability of their application systems. Haiyang Hu [4] presented a new approach to evaluate the reliability of the component based software systems in the open distributed system environment by analyzing the reliabilities of the components of different application domains, the reliabilities of the connections to these components and the architecture style of their composition. Jung-Hua Loet al. [7] proposed a new approach to analyzing the reliability of the system, based on the reliabilities of the individual components and the architecture of the system for predicting the reliability of component based software systems.

Sherif M.Yacoub et al. [15] presented a methodology for reliability risk assessment at the early stages of the development lifecycle, namely the architecture level. They described a heuristic risk assessment methodology that is based on dynamic metrics. The methodology uses dynamic complexity and dynamic coupling metrics to define complexity factors for the architecture elements. Severity analysis is performed using Failure Mode and Effect Analysis (FMEA) as applied to architecture models. William W.Everett [20] described an approach for analyzing the reliability of software systems using component analysis. It uses the Extended Execution Time (EET) reliability growth model at the software component level. Jeffrey M. Voas [6] introduced a methodology for determining the quality of off-the-shelf (OTS) components using a set of Black-Box analysis. This methodology provides the useful information for software developers for choosing components and for defending themselves legally against someone else's imperfect OTS components Denise M.Woit et al. [3] presented a set of component design and interaction rules which, if followed in software development, can produce systems with the highly independent components necessary in order to legitimately calculate system reliability from component reliability. Stochastic independence is a fundamental requirement of calculating system reliability from component reliabilities. Denise M.Voit et al. [2] showed how to use the Continuation Passing Style to covert conventional functions into fragments that can be used in building Markov models. Saileshwar Krishnamurthy et al.[14] reported an experiment to evaluate a method, known as Component Based Reliability Estimation (CBRE), for the estimation of reliability of a software system using reliabilities of its components. CBRE involves computing path reliability estimates based on the sequence of components executed for each test input. Path reliability estimates are averaged over all test runs to obtain an estimate of the system reliability. With the clarity of above works this paper proposes a graph theoretical approach for listing the execution paths and a reliability model for component based software systems from the execution path of software components connected in series, parallel or mixed configuration network structure.

3.1 Algorithm for the Proposed Model

Input : Connectivity Matrix and Adjacency Matrix

Output : Reliability Estimation of Software Component

(i) Begin

(ii) Draw the Software Component flow graph from the flowchart using the procedure mentioned in Section 3

(iii) Number the nodes and branches of the Software Component flow graph arbitrarily

(iv) Decompose the entire Software Component flow graph into small sub graphs

(v) Construct the connection matrix (C) of order n x n, where n is the number of nodes, as follows

$$(C) = \begin{cases} C_{ij}, \text{ if the branch (i,j) exists between} \\ \text{nodes i and j} \\ 0, \text{ Otherwise} \end{cases}$$

(vi) Construct column vectors V(k), k=1,2,…….n , scanning the corresponding columns of [C] for each row where elements of each V(k) are the row numbers which contains non-zero entries.

For example the connection matrix [C] is defined as follows

$$[C] = \begin{pmatrix} 0 & 1 & 0 & 2 & 0 \\ 1 & 0 & 6 & 4 & 0 \\ 0 & 6 & 0 & 5 & 3 \\ 2 & 4 & 5 & 0 & 7 \\ 0 & 0 & 3 & 7 & 0 \end{pmatrix}$$

The five column vectors of the index are prepared by scanning columns 1-5 of (C). In the above matrix C_{21} and C_{41} are non-zero elements in the column1. Hence, in V (1), 2 and 4 are replaced. In Column2, C_{12}, C_{32}, and C_{42} are non-zero entries. So 1, 3 and 4 are placed in V(2). All other columns of the index are prepared in the similar manner.

Hence the index is

V(1)	V(2)	V(3)	V(4)	V(5)
2	1	2	1	3
4	3	4	2	4
	4	5	3	
			5	

vii) Construct the first order row vector R(1) taking the first row of (C)

viii) Obtain the second order row vector R(2) by multiplying R(1) and column vector V(k) using the method given in [10]. Continue this procedure to construct the other higher order row vectors and finally obtain all feasible paths of each sub-graph of the software component flow graph

ix) Obtain the reliability expression for all feasible paths for each sub graph using Abraham's[1] method of disjointing

x) Combine the reliability models of all the sub graphs to obtain the reliability model for the entire software component flow graph

(xi) End

IV. ILLUSTRATIONS

4.1 Software Component for Binary Search *(C++ Source)*

```
void Binary_Search(elem key,elem *T,int size,boolean
&found,int &L)

        {

        int bott,top,mid;

        bott=0;

        top=size-1;

        L=(top+bott)/2;

        if(T(L)==key)

                found=true;

        else    found=false;

                while(bott<=top && !found)

                {

                        mid=top+bott /2;

                        if(T[mid]==key)

                        {

                                found=true;

                                L=mid;

                        }

                        else if (T[mid]<key)

                                bott=mid+1;

                        else

                                top=mid-1

                }       }
```

4.2 Software Component Flow graph for the Binary Search Routine

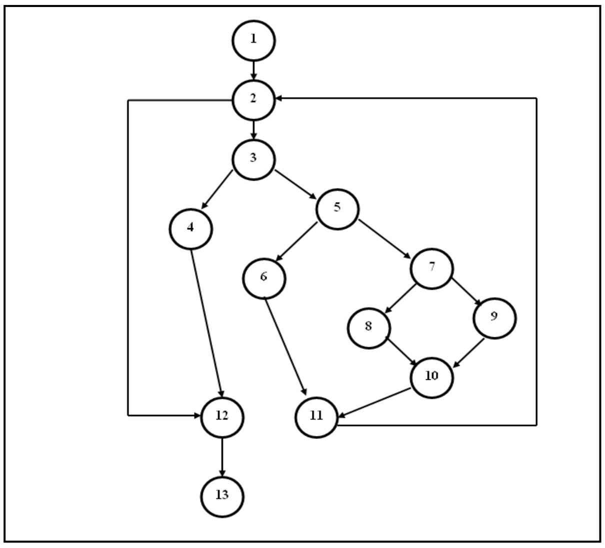

Figure 3. Binary Search Component Flow Graph

4.3 Independent Execution Paths

Path 1 : 1,2,12,13

Path 2 : 1,2,3,4,12,13

Path 3 : 1, 2, 3,5,6,11,2,12,13

Path 4 : 1,2,3,5,7,8,10,11,2,12,13

Path 5 : 1,2,3,5,7,9,10,11,2,12,13

4.4 Adjacency Matrix

$$
(C) = \begin{array}{c|ccccccccccccc}
 & \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} & \mathbf{7} & \mathbf{8} & \mathbf{9} & \mathbf{10} & \mathbf{11} & \mathbf{12} & \mathbf{13} \\
\hline
\mathbf{1} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{2} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\mathbf{3} & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\mathbf{5} & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\mathbf{7} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
\mathbf{8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\mathbf{9} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\mathbf{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\mathbf{11} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\mathbf{13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

4.5 Path Matrix

$$(C) = \begin{array}{c|ccccccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\ \hline 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 3 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 11 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

From the path matrix the reliability model of the logical flow of the software component flow graph is obtained using the Disjoint Algorithm [1]. The disjoint algorithm first converts the complex and all the series parallel network structures in series using idempotence rule and the reliability model is estimated. The reliability model for logical flow for the binary search software component is as follows

$$R_d = p_1p_2p_{15} + p_1q_2p_3p_4p_{14}p_{15} + p_1q_2p_3p_5p_6p_{13}p_{16}p_2p_{15} + p_1q_2p_3p_5p_7p_8p_{10}p_{12}p_{16}p_2p_{15} + p_1q_2p_3p_5p_7p_9p_{11}p_{12}p_{16}p_2p_{15} \quad (16)$$

VII. CONCLUSION

The presented approach helps in estimating the reliability of component based software systems by determining all the independent execution paths of the software component. In brief, the algorithm used for path enumeration avoids unnecessary multiplication and addition of zeros in the adjacency matrix thereby saving computation time. Also, the disjointing method reduces number of terms in the overall reliability model which leads to the appreciable reduction in computation time and also it increases the probability of obtaining more accurate result. This method gives a detailed description of the factors which are vital for the design and development process of component based software systems.

REFERENCES

1. Abraham, J. A., , An improved algorithm for network reliability, IEEE Transactions on Reliability,28 (1979),pp. 58-61.
2. Denise M. Woit, David V. Mason. Software Component Independence. HASE, (1998),pp.74-81.
3. Dick Hamlet, Denise M.Woit and David V.Mason,"Theory of software reliability based on components", ICSE '01 Proceedings of the 23rd International Conference on Software Engineering,(2001), pp.361 – 370.
4. Haiyang Hu ,Reliability Analysis for Component-based Software System in Open Distributed Environments, IJCSNS International Journal of Computer Science and Network Security, 7;5;(2007), pp.193-202.
5. J.D. Musa, A. Iannino, and K. Okumoto. Software Reliability: Measurement, Prediction, Applica- tion. McGraw-Hill, 1987
6. Jeffrey M. Voas, "Certifying Off-the-Shelf Software Components," Computer, 31;6,(1998) ,pp.53-59.
7. Jung-Hua Lo, Chin-Yu Huang, Sy-Yen Kuo, Michael R. Lyu , "Sensitivity Analysis of Software Reliability for Component-Based Software Applications, 27th Annual International Computer Software and Applications,2003.
8. K. F. Fischer and M. G. Walker, "Improved software reliability through requirements verification," IEEE Trans. Rel., vol. R-28, pp. 233-240.
9. LEVESON, N. G., AND STOLZY, g.' L. Safety analysis of Ada programs using fault trees. IEEE Trans. Reliability R-32, 5 (1983), pp.479-484.
10. M. A. Aziz, M. A. Sobhan and M. A. Samad, Reduction in computations in enumeration of terminal and multiterminal pathset by the method of indexing, Micro electron. Reliab., 32;8,(1992), pp. 1067-1072.
11. Norman F. Schneidewind, "Application of Program Graphs and Complexity Analysis to Software Development and Testing, IEEE Transactions on Reliability, vol. R-28, no.3, August (1979), pp. 192-198.
12. P K Suri and Sandeep Kumar, Design of Simulator for Reliability Estimation of Component Based Software System IJCSNS International Journal of Computer Science and Network Security,9;9,(2009), pp.161-167.
13. R.B.Misra and K.B.Misra, Enumeration of all simple paths in a communication network,. Microelectronics Reliability. Vol. 20, pp. 419-426
14. Saileshwar Krishnamurthy Aditya P. Mathur, On theEstimation of Reliability of a Software System

Using Reliabilities of its Components- IEEE (1997), pp.146-155.

15. Sherif M.Yacoub and Hany H.Ammar “A Methodology for Architecture-Level Reliability Risk Analysis”, IEEE Transactions on Software Engineering, 28;6,(2002), pp.529-547.

16. Shooman, Martin L., Software Engineering - Design,. Reliability and Management, McGraw-Hill: New York, 1983.

17. Suri, P.K. and Aggarwal, K.K., “Software Reliability of Programs with Network Structure,” Microelectron Reliability, 21;2, (1981), pp. 203-207.

18. Suri.P.K. and Aggarwal.K.K. "Reliability evaluation of computer program,"Microdectronics and Reliability, 20, 1980.

19. Tirthankar Gayen and R. B. Misra, Reliability Bounds Prediction of COTS Component Based Software Application, International Journal of Computer Science and Network Security, 8;12,(2008), pp.219-228.

W. Everett, “Software Component Reliability Analysis”, Proc. Symp. Application– Specific