An Experimental Study on Reliability Estimation of GNU Compiler Components – A Review

R.Chinnaiyan
Associate Professor
Department of CA
AVC College of Engineering.

Dr.S.Somasundaram
Associate Professor
Department of Maths
CIT –Coimbatore.

ABSTRACT

In today's software arena most of the software's are developed using the component based software development methodology. Hence it is necessary to ensure that the developed software possess high reliability as perceived by the consumer before the software release. Many researchers have proposed various analytical models for assessing the reliability of component based software systems, where the black-box testing procedure is used for reliability assessment for evaluating the software components where internal workings of the software components are not evaluated. So it is necessary for the software developers to use white box testing technique for ensuring high reliability of the software components which yields reliable software system. This paper proposes a renewed reliability engineering approach and it is elucidated with the real software system case study with GNU compiler.

Keywords: Software Components, Component Based Software Reliability, GNU Compiler

1. INTRODUCTION

IEEE 610.12-1990 defines reliability as "The ability of a software system or component to perform its required functions under stated conditions for a specified period of 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 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. Additionally, their relation to dependability attributes of the modeled software system is unknown.

1.1 Background

P K Suri et al. [11] 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 any other component as per the Markov process. Tirthankar Gayen et al.[17] 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 bound on the reliability of their application systems. Haiyang Hu [13] 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. [8] 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 assessing the reliability of component based software system.

Sherif M.Yacoub et al. [16] 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 [18] described an approach for analyzing the reliability of software systems using component analysis. It uses the Extended Executing Time (EET) reliability growth model at the software component level. Jeffrey M. Voas [7] 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.

Independence is a fundamental requirement of calculating system reliability from component reliabilities. Denise M.Voit et al. [2] showed how to use the CPS transformation to covert conventional functions / procedures into fragments that can be used in building markov models. Saileshwar Krishnamurthy et al.[15] 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 renewed reliability engineering approach for estimating the reliability of component based software system and it is elucidated with the real time case study with GNU compiler.

2. DESCRIPTION

2.1 GNU Compiler

GCC is the acronym for GNU Compiler Collection. GCC is an integrated distribution of compilers for several high levels programming languages such us C, C++, Objective-C, Java, FORTRAN, and Ada. Front-End is also a compiler which is specific to a particular language. GCC supports front-ends like Pascal, Mercury and COBOL in addition the above mentioned languages. Initially it was referred to as GNU C compiler when it was used only to compile C programs. The majority of the compiler optimizers are included in the language independent component of GCC. It also includes all the 'back-ends', which are used to generate machine code for various processors. GCC is Free Open Source Software. Software Components are the part of GCC compiler that is devoted to a specific functionality. GCC compiler has a number of software components. For elucidating the proposed approach only 12 software components of the GCC compiler are taken. The 108 files of the compiler are mapped into 12 software components to build the design of GCC.

2.2 Size of GNU Compiler

GNU Compiler is the leading case study ever used for experimental software reliability estimation. C proper part was used in the proposed experimental approach of GNU Compiler. The C proper part itself has 300 source files written in 12 different languages and has approximately 800,000 lines of ANSI C code. These files include both the programming and scripting languages. Table 1 contains the list of languages and lines of code written in each of those languages for the version GCC-2.96-20000731. This table shows how large the case study actually is.

Table 1: List of Languages with Lines of Code of GCC Compiler

Language	LOC
ANSI C	789,901
CPP	126,738
YACC	19,272
SH	17,993
ASM	14,559
LISP	7,161
FORTRAN	3,814
EXPECT	3,705
SED	310
PERL	144
OBJC	479
Total	984,076

3. RELIABILITY OF SOFTWARE COMPONENTS

"The reliability of Software Component ***i*** is the probability ***Ri*** that the Software component performs its function correctly". If the number of failures of each software component is very small when compared to the number of executions of each software component, then the reliabilities of software components must be very high. There are many methods to calculate the reliabilities of software components. The historical data and the requirement documents are used during the early stages of development. Software Reliability growth models for each software component are used [18]. However, the software component failure data available is not sufficient to apply these models. The information about the succeeded and failed executions during the testing are used to find the reliabilities of software components [1], [4], [6], [10].

These methods depend heavily on the type and nature of the test cases used to find the software faults. Irrespective of the method used to find out the reliabilities of software components, the values may be erroneous. The mean value of reliability for each software component is then estimated. The reliability of a software component is assessed using the Equation (2)

$$Qi = \lim_{ni \to \infty} (fi / ni) \longrightarrow (1)$$

$$Ri = 1 - Qi \longrightarrow (2)$$

Where fi is the number of failures and n_i is the number of executions of software component i in N randomly generated test cases. ***Ni*** is total number of times the software component is executed in 2126 test cases. Qi is the unreliability of the Software Component and *Ri* is the reliability of the software component *i.* The reliabilities of software components are extremely high and almost equal to one. This is because only few failures are compared to the number of executions of each software component.

4. NUMERICAL ILLUSTRATIONS

The information available in the documentation of GNU Complier is not sufficient for dividing the entire software system into constituent software components. More than 50 files out of 108 files are missing from the documentation. The source codes are examined for understanding what each file does and assign that file to the appropriate software component. Based on unique functionalities the GNU Compiler is divided into 12 software components. Files are assigned to each software component on the basis of their functionality. The software components and the number of files in each software component are shown in Table.2

Table 2. Software Components and Number of Files in each Software Component of GNU Compiler

Software Component Name	Component ID	No. of Files in each Software Component
Parsing	1	32
Tree Optimization	2	11
RTL Generation	3	26
Jump Optimization	4	4
CSE	5	4
GCSE	6	2
Loop Optimization	7	10
Register Allocation	8	11
Branch Processing	9	4
Final Pass	10	9
Library Files	11	21
Top Level Control	12	1

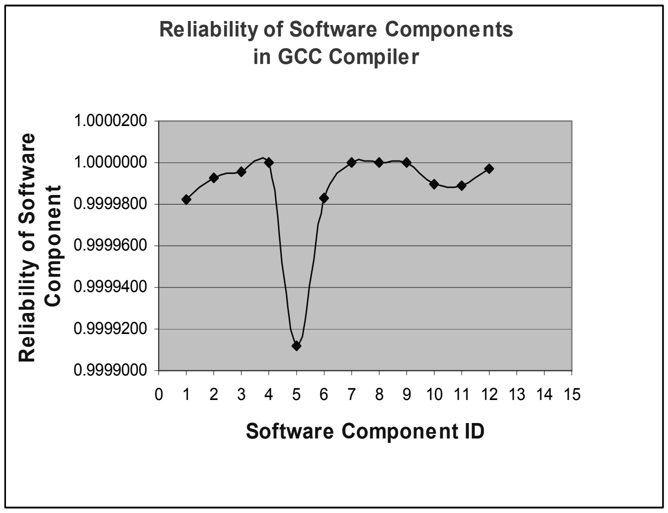

Figure 1. Reliability of Software Components in GNU Compiler

Software Component ID	Number of Failures (Fi)	Number Of Executions (Ni)	Reliability of Software Component (Ri)
1	30	1656221	0.9999819
2	1	135180	0.9999926
3	7	1688076	0.9999959
4	0	162338	1.0000000
5	1	11326	0.9999117
6	1	57377	0.9999826
7	0	72680	1.0000000
8	0	372486	1.0000000
9	0	16087	1.0000000
10	4	381046	0.9999895
11	10	919668	0.9999891
12	1	302504	0.9999967

Table 3. Reliability of Software Components of GNU Compiler

5. RELIABILITY OF COMPONENT BASED SOFTWARE

The state-based composite method is proposed by R.C. Cheung in [12] to combine the software architecture with the failure behavior of the component based software systems. The model assumes a single entry node and single exit node for the software system. In this proposed approach two absorbing states C and F are added to the Discrete Time Markov chain (DTMC). These states represent the successful completion and failure of the software system respectively. Two dummy states BEGIN and END were already added in the operational profile which represents the beginning and ending of the software execution. The transition probability P is converted to P1. The transition probability P_{ij} in the original matrix is then converted to R_iP_{ij} to generate the values in P_1. *Ri* is the reliability of the software component *i*. R_iP_{ij} is the probability that the software component i produces the correct result and the control is transferred to software component j [14]. An arc is made between the failure state and the software component i with a transition probability of $(1 - R_i)$, to consider the failure of software component i. The software components C and F are not considered when calculating the entire software system reliability. The reliability of the component based software system is the probability that the control reaches state C from BEGIN state. The matrix P1 is converted into Q by deleting the rows corresponding to C and F. The element *Qk* (1, n) represents the probability of reaching state n from BEGIN state with k transitions [14]. The number of transitions ranges from 0 to infinity. So the reliability of component based software system is $R = S$ (1, n) *Rn*. MATLAB is used to implement the equation to find the reliability of the software system.

The value for the reliability calculated using this method is 0.9201. The reliability of the software system is also calculated using the black box testing and the two values were compared. The error in the reliability estimate is given by the following equation $| (R_{model} - R_{actual}) / R_{actual} | \times 100$. $| (0.9201 - 0.9741) / 0.9741 | \times 100 =$ **5.5 %**

The error in estimation is a mere 5.5%.

6. CONCLUSION

This proposed approach presented an experimental approach for analyzing the reliability component based software. The GNU compiler and its component failures dataset is used for experimentation. This is one of the leading case studies ever used for reliability analysis of component based software systems. The most important thing that distinguishes the proposed work from most of the related works is the size of the case study that is used. The problems associated with the experiments on the studies are explained. Most of the potentially difficult problems associated with large-scale software applications are addressed. All previous studies on experimental studies mentioned in the related work, contributed to a small set of these problems. The reliability for each software component is calculated with the failures of software components during execution. The reliability of component based software system is thus calculated using both black-box method and the white-box method. Accurate value for the reliability is arrived, with only 5% of difference between the values found using the two methods.

7. REFERENCES

[1] B.Littlewood and D.Wright, "Some Conservative Stopping Rules for Operational Testing of Safety – Critical Software" IEEE Trans. Software Engineering, 1993; 23;11:673-683.

[2] Denise M. Woit, David V. Mason. Software Component Independence. HASE'1998:74-81.\

[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,2001:361 – 370.

[4] E. Nelson, "A Statistical Bases for Software Reliability", TRW-SS-73- 02, TRW Software series,1973.

[5] Bugzilla available at: http://www.bugzilla.org.

[6] J.H.Poore, H.D.Mills and D.Mutchler, "Planning and Certifying Software System Reliability", IEEE Software, 1993:88- 99.

[7] Jeffrey M. Voas, "Certifying Off-the-Shelf Software Components," Computer, 1998;31;6: 53-59.

[8] 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.

[9] K. Goseva Popstojanova and Sunil. K. Kamavaram, "Assessing Uncertainty in Reliability of Component-Based Software System", Proc. 14th IEEE International Symposium on Software Reliability (ISSRE 2003), Denver, CO, 2003.

[10] K.W. Miller, L. J. Morell, R. E. Noonan, S. K. Park, D. M. Nikol, B.W. Murrill, and J. M. Voas, "Estimating the Probability of Failure when Testing Reveals no Failures", IEEE Trans. Software Engineering, 1992;18;1: 33- 43.

[11] 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, 2009;9;9:161-167.

[12] R. C. Cheung, "A User-Oriented Software Reliability Model", IEEE Trans. Software Engineering, 1980; 6; 2:118-125.

[13] Haiyang Hu ,Reliability Analysis for Component-based Software System in Open Distributed Environments, IJCSNS International Journal of Computer Science and Network Security, 2007;7;5:193-202**.**

[14] S. K. Kamavaram and K. Goseva Popstojanova, "Entropy as a Measure of Uncertainty in Software Reliability", Proc. 13th International Symposium Software Reliability Engineering, Supplementary proceedings ,2002: 209-210.

[15] Saileshwar Krishnamurthy Aditya P. Mathur, On theEstimation of Reliability of a Software System Using Reliabilities of its Components- IEEE 1997: 146-155.

[16] Sherif M.Yacoub and Hany H.Ammar "A Methodology for Architecture-Level Reliability Risk Analysis", IEEE Transactions on Software Engineering, 2002;28;6:529-547.

[17] Tirthankar Gayen and R. B. Misra, Reliability Bounds Prediction of COTS Component Based Software Application, International Journal of Computer Science and Network Security,2008; 8;12 :219-228.

[18] W. Everett, "Software Component Reliability Analysis", Proc. Symp. Application– Specific Systems and Software Engineering Technology, 1999:204-211.

8. AUTHOR'S BIOGRAPHY

1. **R.Chinnaiyan** is working as Associate Professor in the Department of Computer Applications, A.V.C College of Engineering, Mannampandal, Mayiladuthurai , Tamil Nadu , INDIA. He is having 10 years of teaching experience. He is a life member of ISTE and CSI of INDIA. He is now doing his research in Anna University- Chennai at Coimbatore Institute of Technology, Coimbatore. His research interest includes Reliability of Component Based Software, Object Oriented Analysis and Design, Qos and System Reliability.

2. **Dr. S.Somasundaram** is working as Associate Professor in the Department of Mathematics, Coimbatore Institute of Technology, Coimbatore - 641 014, Tamil Nadu, INDIA. He is having 28 years of teaching experience. He is a life member of ISTE, RMS and IMS of INDIA . He had guided over 5 M.Phil Candidates in Bharathiyar and Annamalai Universities. Now he is guiding 5 Ph.D Scholars under Anna University Chennai and Coimbatore His research interest includes Risk Analysis, Qos , Optimization , Networking and Software Reliability