Journal of Information Engineering and Applications
ISSN 2224-5782 (print) ISSN 2225-0506 (online)
Vol.4, No.3, 2014

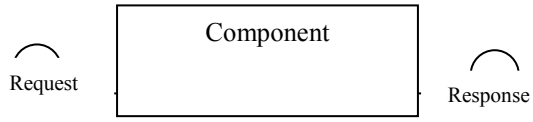
Quality Assessment of Software Reliability Growth Models

Chandra Mouli Venkata Srinivas Akana
Research Scholar, Dept. of CSE, JNTUK, Kakinada, Andhra Pradesh, India
Email: mouliac@yahoo.com
Dr. C. Divakar
Principal, Pydah College of Engineering & Technology, Visakhapatnam, AP, India
Email: divakar_c@yahoo.com
Dr. Ch. Satyanarayana
Professor, Dept of CSE, JNTUK, Kakinada., AP., India.
Email: chsatyanarayana@yahoo.com

Abstract
The two main important properties of software components are reliability and robustness. Reliability can be defined as the probability of failure free operation and on the other hand the robustness can be defined as how far the software can be able to with stand for intrusion attacks. In both the cases there should be some metric to evaluate the performance of these properties. In this paper metrics are been described which can be used to assess the quality of performance for these properties within a software reliability growth model.
Keywords: Software growth model, reliability, robustness, Quality metrics

I.INTRODCUTION

A software component model is a minimal software item for which a separate specification is available [1]. Component-based software engineering (CBSE) seeks to build software systems by composition of pre-existing software components. Two related problems in CBSE are:
(1) How to predict the properties of the assembled system given the properties of individual components and
(2) How to guarantee that one component can be substituted for another without changing the properties of the overall system
Software failures are caused by errors that arise due to faults in the software that have been introduced during software development [5]. A delayed failure is defined for the present research as a failure that occurs some time after the condition that causes the failure and the proposition that delayed failure of software components is a consequence of long-latency error propagation and that the conventional approach to software testing is unlikely to detect this type of software failure
In this paper metrics are been described to characterise a given type of software component and use these metric for the performance evaluation of software reliability growth models.
Software component:
This provides the interface between request and response interfaces see below figure 1



An input to the component arrives in a request and the corresponding output from the component appears in a response. An input to the component may be either valid or invalid; specified component behaviour is to produce a correct response to a valid input and to reject an invalid input.
This paper is organized as follows ,in section I a basic introduction for quality assessment is discussed, in section II previous done work with some examples are quoted and in section III describing about the metric and ending with results and conclusion in section Iv and V .

II.RELATED WORK

Mukherjee *et.al* in [8] presents robustness benchmarks for Unix-like operating systems and gives general design criteria for robustness benchmarks. Slutz *et.al* in [9] describes a system for random generation of SQL (RAGS). This system combines a random testing approach with grammar–based generation to generate large numbers of valid SQL statements; this was successfully used for testing SQL database systems.As stated in [6] shooman's model provides an equation to estimate the expected number of software failures (n_f)

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

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

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

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.RELIABILITY QUALITY METRICS

Reliability can be defined as the probability of failure free operation under stated conditions for specific period of time [2].Assessment of reliability performance for a component are usually defined for the expected input profile in actual operational use.

The commonly used metric for assessment are, Mean time to time failure (MTTF), mean time between failures (MTBF) and robustness [3].In [4] storey has given the definition as a function of time *R(t)* at a constant failure rate of λ

$$R(t) = e^{-\lambda t} \quad (4)$$

Where λ is the probability that there is no failure before time t

Then the MTTF can be given as

$$MTTF = \frac{1}{\lambda} \quad (5)$$

And $MTBF = MTTF + MTTR$ (6)

Where MTTR is the mean time of recovery defined as the average time a component takes to recover from a failure. The measures MTBF, MTTF and MTTR are usually considered to apply in the case of a system operating continuously; however for a system operating on demand as is the case here, equivalent definitions apply where time is treated in discrete units [7].

IV RESULTS AND ANALYSIS

For the experimental analysis a test was conducted on 10k randomly generated statements .At initial state the database was empty creation of tables proceeded till all the tables defined in the input profile has been created for certain runs.

Table I: Tabulated values of acceptance and Error

Run	Executed	Error	Accepted	Succeeded
3	11,682	1	10,135	1,745
6	21,602	8	17,267	3,135
10	21,625	7	18,420	3,159
11	27,252	7	21,840	3,209
12	41,458	11	37,555	5,525
16	12,415	2	12,383	1,220
17	16,152	3	15,956	1,435
19	32,245	8	22,713	3,407
Total	**1,84,431**	**47**	**1,56,269**	**22,835**
Mean	**23,054**	**5.8**	**19,534**	**2,854**

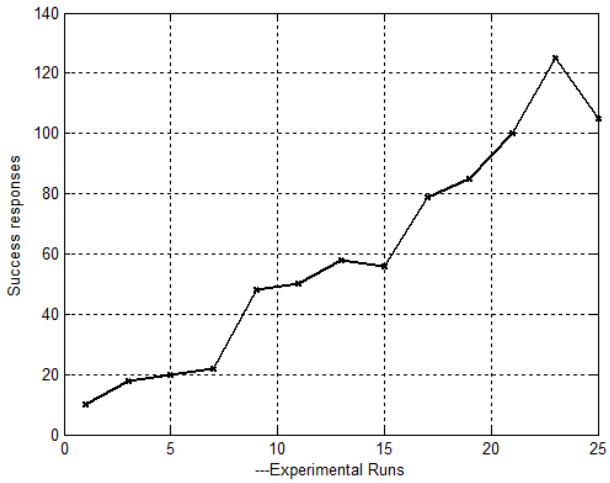


Figure 1: Performance analysis of the component profile

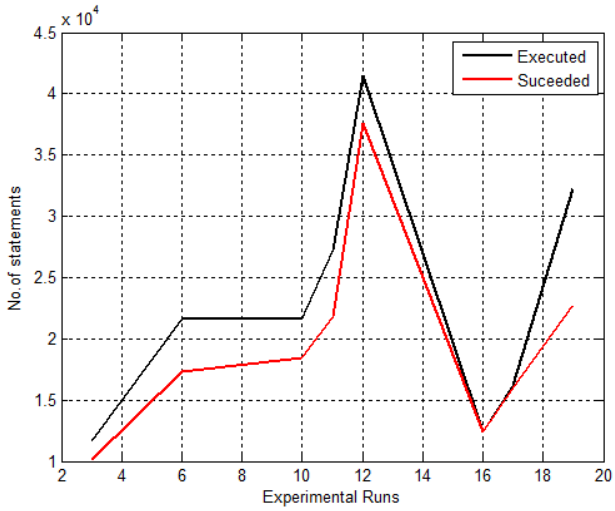


Figure 2: Performance analysis of the component model for the given statements which are executed and succeeded

V.CONCLUSION

Software reliability quality assessment is a very useful concept in evaluating the performance of reliability and robustness of the given component model. The benchmark input profile can be generated and executed automatically in the case of a DBMS using structured query language for which a specification is available. Hence it can be concluded that the MTTF metric can be used to evaluate the performance of both reliability and robustness of the given software component model under any growth model.

REFERENCES

[1] BCS Specialist Interest Group in Software Testing (2004), "Working draft of BS 7925-1 Glossary of terms used in software testing", version 6.3.
[2] Glossary Working Party of the International Software Testing Qualification Board (ISTQB), van Veenendaal, E. (ed.) (2006), Standard glossary of terms used in Software Testing, Version 1.2.
[3] Standards Coordinating Committee of the IEEE Computer Society (1991), IEEE standard computer dictionary 610.
[4] Storey, N (1996), Safety-Critical Computer Systems, Prentice Hall.
[5] Ammann, P, Offutt, J, Introduction to Software Testing, Cambridge University Press, 2008
[6] 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.
[7] Fenton, N, Pfleeger, S (1997), Software Metrics: A Rigorous & Practical Approach (2nd edition), PWS Publishing Company
[8] Mukherjee, A, Siewiorek, D (1997), "Measuring Software Dependability by Robustness Benchmarking", IEEE Transactions on Software Engineering.
[9] Slutz, D (1998), "Massive Stochastic Testing of SQL", VLDB'98 Proceedings of 24th International Conference on Very Large Data Bases.
[10] Binder, R.V, Testing Object Oriented Systems: Models Patterns and Tools, Addison-Wesley, 2000.
[11] Binder, R.V, "Automated Testing with an Operational Profile", DoD Software Tech News, Volume 8, Number 1, 2004.
[12] Chen, T.Y, Merkel, R (2007), "Quasi-Random Testing", IEEE Transactions on Reliability, Vol. 56, No. 3.