Markovian Reliability Analysis for Software using Error Generation and Imperfect Debugging

[1]M. Kaushik, and [2]G. Kumar

Abstract—**This investigation deals with a markovian analysis for software reliability model using errors generations and imperfect debugging. Three types of errors are taken into consideration for developing a software reliability model. The debugging is done in a manner without distinguishing between the three types of errors. Moreover, Runge-Kutta (RK) method of fourth order is applied for analyzing the software reliability of different configurations under transient condition. Various characteristics of software reliability assessment are suggested. The effect of different parameters on system performance indices are demonstrated graphically.**

Keywords—**Software reliability, error generation, imperfect debugging, markov model, R-K method .**

I. Introduction

SOFTAWRE reliability is probably the most important feature distressing the software quality. Software quality is measured by counting the number of faults in the program or software. Software reliability helps to the software developers and users for increasing the system efficiency. To identify and eliminate errors in software development process and also to improve software reliability, the software reliability analysis is highly recommended [1]. Several reliability models have been used for Markov chain-based testing [2]. Software reliability plays an important role in assuring the quality of software [3]. Software reliability testing is concerned with the quantitative relationship between software testing and software reliability [4]. To estimate the reliability and software failures through mathematical expression, software reliability growth model has been used [5].

Software reliability is a major troubled state of mind in various organization. After release of software, software shows some minor or major bugs. In real practice of software development, the number of failure removed during development phase need not to be same as the number of faults observed. A system is said to have a failure if the service it deliver to the user deviates from compliance with the system specification for a specified period of time. Reason for failure may be software or hardware failure. A software failure is the departure of the external results of program operation from requirements [5-6]. Consider it with example as a user requests an operation at program start-up. It does not display. In general, software reliability is directly allied with software error, defects and failure. In our investigation, the software may fail due to simple error, complex error and critical error.

Due to the complexity in software systems, testing engineer may not able to remove all faults perfectly, and the original fault may remain exist which termed as ***imperfect debugging*** (ID). In the case of ID, if new bugs are introduced during debugging or the bug that caused the failure is not successfully removed; [7] while in the case of error generation, the fault content increases as the testing progresses. New faults [8] may introduce by removal of observed faults. A Markov model was suggested to explore the quantitative relationships between software testing and software reliability in the presence of imperfect debugging [9]. Moreover, the debugging process is usually far from perfect and actually many faults encountered by customers are those introduced during debugging [10].

In this paper, a framework is proposed to develop a markovian software reliability model with three types of errors and imperfect debugging. The rest of this paper is organized as follows. In section II, the assumptions and notations are given to develop the mathematical model. In section III, governing equations are derived based on different assumptions of fault introduction and the correction effort. In section IV, several performance measures are given. In section V, numerical results are presented. Conclusion has been given in section VI.

II. Mathematical Model

In this section, we develop the Markov model for the software having three types of errors. For formulating the model, we define a random variable representing the cumulative number of errors successfully covered upto a certain level of time. Every software shows some bugs called errors after being released. The software may fail either due to simple error, complex error and critical error. The maximum number of errors of all types in the software are never exceeds upto a finite limit say 'N'. When a failure occurs, an instantaneous repair effort start. The repairman is always available for removing the errors. Debugging is imperfect in the software. The life times and repair times of errors are exponentially distributed with constant rates.

Notations:

M. Kaushik is with the department of computer science & engineering, JECRC University, Jaipur-303905 India (email: manju.kaushik@jecre.edu.in)

G. Kumar is with the department of computer science & engineering, JK Lakshmipat University, Jaipur-302026, India (phone:141-710-7581: email: gireesh8@gmail.com)

N	Maximum number of faults contain in the software
$\alpha\ (\beta, \gamma)$	Failure rate of a software due to simple error, complex error and critical error
$\alpha'\ (\beta', \gamma')$	Repair rate of a software when it fails due to simple error, complex error and critical error
p_2	Probability that the simple error occurs in the software
p_0	Probability that the simple error removed from the software
q_2	Probability that the complex error occurs in the software
q_0	Probability that the complex error removed from the software
r_2	Probability that the critical error occurs in the software
r_0	Probability that the critical error removed from the software
(i, j, k)	Triplet denoting the number of errors due to simple error, complex error and critical error respectively
$P_{(0,0,0)}(t)$	Probability that there is no error in the software at time
$P_{(i,j,k)}(t)$	Probability that there are i ($0 \leq i \leq N$), j($0 \leq j \leq N$) and k($0 \leq k \leq N$) errors at time t in the software due to SE, CE and CRE**,** respectively

III. Governing Equations

In this section, we construct the differential difference equations governing the Markov model of software having three types of errors and imperfect debugging which are given as:

$$\frac{d}{dt}P_{0,0,0}(t) = \alpha' p_0 P_{1,0,0}(t) + \beta' q_0 P_{0,1,0}(t) + \gamma' r_0 P_{0,0,1}(t) \tag{1}$$

$$\frac{d}{dt}P_{i,0,0}(t) = -(i\alpha p_2 + i\alpha' p_0)P_{i,0,0}(t) + \beta' q_0 P_{i,1,0}(t) + \gamma' r_0 P_{i,0,1}(t) + (i+1)\alpha' p_0 P_{i+1,0,0}(t) + (i-1)\alpha p_2 P_{i-1,0,0}(t), \qquad 1 \leq i \leq N-1 \tag{2}$$

$$\frac{d}{dt}P_{N,0,0}(t) = -N\alpha' p_0 P_{N,0,0}(t) + (N-1)\alpha p_2 P_{N-1,0,0}(t) \tag{3}$$

$$\frac{d}{dt}P_{0,j,0}(t) = -(j\beta q_2 + j\beta' q_0)P_{0,j,0}(t) + \alpha' p_0 P_{1,j,0}(t) + \gamma' r_0 P_{0,j,1}(t) + (j+1)\beta' q_0 P_{0,j+1,0}(t) + (j-1)\beta q_2 P_{0,j-1,0}(t), \qquad 1 \leq j \leq N-1 \tag{4}$$

$$\frac{d}{dt}P_{0,N,0}(t) = N\beta' q_0 P_{0,N,0}(t) + (N-1)\beta q_2 P_{0,N-1,0}(t) \tag{5}$$

$$\frac{d}{dt}P_{0,0,k}(t) = -(k\gamma r_2 + k\gamma' r_0)P_{0,0,k}(t) + \alpha' p_0 P_{1,0,k}(t) + \beta' q_0 P_{0,1,k}(t) + (k+1)\gamma' r_0 P_{0,0,k+1}(t) + \gamma r_2 P_{0,0,k-1}(t), \qquad 1 \leq k \leq N-1 \tag{6}$$

$$\frac{d}{dt}P_{0,0,N}(t) = -N\gamma' r_0 P_{0,0,N}(t) + (N-1)\gamma r_2 P_{0,0,N-1}(t) \tag{7}$$

$$\frac{d}{dt}P_{i,j,0}(t) = -(i\alpha p_2 + i\alpha' p_0 + j\beta q_2 + j\beta' q_0)P_{i,j,0}(t) + (i-1)\alpha p_2 P_{i-1,j,0}(t) + (j-1)\beta q_2 P_{i,j-1,0}(t) + (i+1)\alpha' p_0 P_{i+1,j,0}(t) + (j+1)\beta' q_0 P_{i,j+1,0}(t), \quad i,j \neq 0, 2 \leq i+j \leq N-1 \tag{8}$$

$$\frac{d}{dt}P_{i,j,0}(t) = -(i\alpha' p_0 + j\beta' q_0)P_{i,j,0}(t) + (i-1)\alpha p_2 P_{i-1,j,0}(t) + (j-1)\beta q_2 P_{i,j-1,0}(t) \qquad i,j \neq 0, i+j = N \tag{9}$$

$$\frac{d}{dt}P_{0,j,k}(t) = -(j\beta q_2 + j\beta' q_0 + k\gamma r_2 + k\gamma' r_0)P_{0,j,k}(t) + (j-1)\beta q_2 P_{0,j-1,k}(t) + (k-1)\gamma r_2 P_{0,j,k-1}(t) + (j+1)\beta' q_0 P_{0,j+1,k}(t) + (k+1)\gamma' r_0 P_{0,j,k+1}(t), \quad j,k \neq 0, 2 \leq j+k \leq N-1 \tag{10}$$

$$\frac{d}{dt}P_{0,j,k}(t) = -(j\beta' q_0 + k\gamma' r_0)P_{0,j,k}(t) + (k-1)\gamma r_2 P_{0,j,k-1}(t) + (j-1)\beta q_2 P_{0,j-1,k}(t) \qquad j,k \neq 0, j+k = N \tag{11}$$

$$\frac{d}{dt}P_{i,0,k}(t) = -(i\alpha p_2 + i\alpha' p_0 + k\gamma r_2 + k\gamma' r_0)P_{i,0,k}(t) + (i-1)\alpha p_2 P_{i-1,0,k}(t) + (k-1)\gamma r_2 P_{i,0,k-1}(t) + (i+1)\alpha' p_0 P_{i+1,0,k}(t) + (k+1)\gamma' r_0 P_{i,0,k+1}(t), i,k \neq 0, 2 \leq i+k \leq N-1 \tag{12}$$

$$\frac{d}{dt}P_{i,0,k}(t) = -(i\alpha' p_0 + k\gamma' r_0)P_{i,0,k}(t) + (i-1)\alpha p_2 P_{i-1,0,k}(t) + (k-1)\gamma r_2 P_{i,0,k-1}(t) \qquad i,k \neq 0, i+k = N \tag{13}$$

$$\frac{d}{dt}P_{i,j,k}(t) = -(i\alpha p_2 + i\alpha' p_0 + j\beta q_2 + j\beta' q_0 + k\gamma r_2 + k\gamma' r_0)P_{i,j,k}(t) + (i-1)\alpha p_2 P_{i-1,j,k}(t) + (j-1)\beta q_2 P_{i,j-1,k}(t) + (k-1)\gamma r_2 P_{i,j,k-1}(t) + (i+1)\alpha' p_0 P_{i+1,j,k}(t) + (j+1)\beta' q_0 P_{i,j+1,k}(t) + (k+1)\gamma' r_0 P_{i,j,k+1}(t), \quad i,j,k \neq 0, 2 \leq i+j+k \leq N-1 \tag{14}$$

$$\frac{d}{dt}P_{i,j,k}(t) = -(i\alpha' p_0 + j\beta' q_0 + k\gamma' r_0)P_{i,j,k}(t) + (i-1)\alpha p_2 P_{i-1,j,k}(t) + (j-1)\beta q_2 P_{i,j-1,k}(t) + (k-1)\gamma r_2 P_{i,0,k-1}(t)\ , i,j,k \neq 0, i+j+k = N \tag{15}$$

IV. PERFORMANCE MEASURE

In this section, we establish various performance indices in terms of transient probabilities. For finding these probabilities, the transient equations (1)-(15) are solved using Runge-Kutta (R-K) method for the software having total four errors of each type. R-K technique is implemented by developing program in MATLAB software. After obtaining transient probabilities, some performance indices are calculated as:

- The probability of perfect program at testing time ‘t’ is calculated as
$$P(T) = P_{0,0,0}(t) \tag{16}$$
- The expected number of faults remaining in the software at testing time ‘t’ is given as
$$F(T) = \sum_{i=1}^{N} i \sum_{j=0}^{N-i} \sum_{k=0}^{N-i-j} P_{i,j,k}(t) + \sum_{j=1}^{N} j \sum_{k=0}^{N-j} \sum_{i=0}^{N-j-k} P_{i,j,k}(t) + \sum_{k=1}^{N} k \sum_{i=0}^{N-k} \sum_{j=0}^{N-k-i} P_{i,j,k}(t) \tag{17}$$
- The software reliability of the system is
$$P(T) = \sum_{i+j+k=1}^{N-1} P_{i,j,k}(t) \tag{18}$$

V. NUMERICAL RESULTS

In this section, we are interested in sensitivity analysis by taking the numerical illustrations. For this purpose, software ‘MATLAB’ is used to develop a computational program and to analyze the system performance numerically. For illustration purpose, we obtain the results for transient reliability for the system having three types of error. The classical R-K method of forth order is implemented by using the “ode45” function. In transient case, the numerical computations based on empirical values of failure and repair parameters have been carried out by taking the time span [0, 5] with equal intervals of 1 units. The results are summarized in figures 1-4. The default parameters are chosen as $\alpha = 0.01, \alpha' = 0.1, \beta = 0.2, \beta' = 0.2, \gamma = 0.03, \gamma' = 0.3, p_0 = 0.1, p_2 = 0.6, q_0 = 0.2, q_2 = 0.6, r_0 = 0.4 \; and \; r_2 = 0.6$.

From figs 1-2, it is observed that the reliability of software increases sharply as the values of testing time attains the higher values. Figs. 1-2 depict that the system reliability shows the decreasing trend with the increasing values of failure rates.

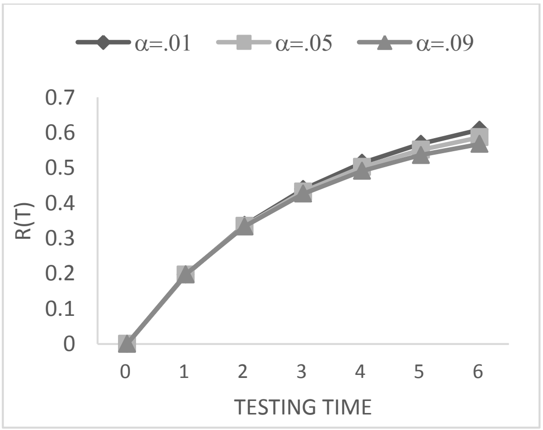

Fig. 1. Reliability vs time by varying α.

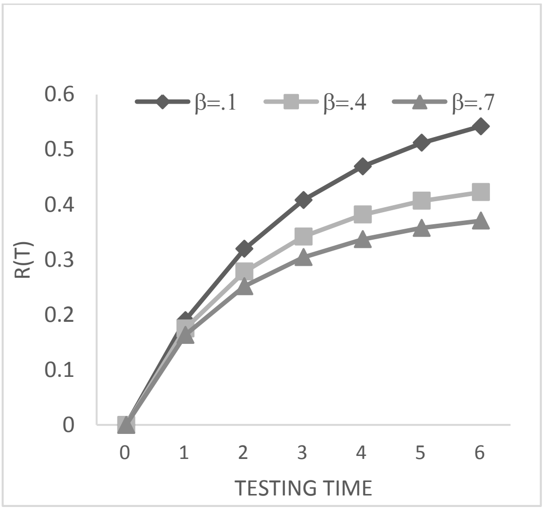

Fig. 2. Reliability vs time by varying β.

From figs 3-4, it is concluded that the mean number of faults decreases as time increases but increase for all increasing values of failure rates.

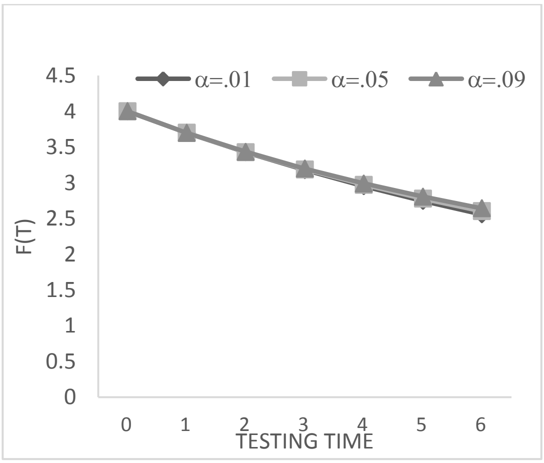

Fig. 3. F(T) vs time by varying α.

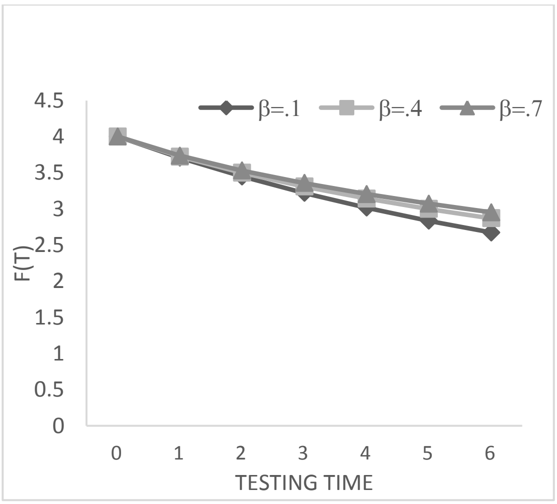

Fig. 4. F(T) vs time by varying β.

Finally, from the above tables and figs, it can be predicted that the software reliability could be increased up to certain time by increasing the repair rates.

VI. CONCLUSION

In this paper, we have developed a markovian software reliability model with three types of errors and imperfect debugging. The suggested model is suitable and helpful in area of reliability engineering. The transient availability and other performance indices obtained may be helpful to improve the software reliability. We have derived the expressions for the system reliability and availability under different configurations.

REFERENCES

[1] Jelinski Z. and Moranda P. B., “Software reliability research”, Statistical Computer Performance Evaluation, W. Freiberger Ed., Academic Press, New York, 465 (1972).

[2] Prowell S. J. and Poore, J. H., “Computing system reliability using markov chain usage models”, *The journal of Systems and Software*, **73**, 219-225, (2004).

[3] Jain M. and Priya K., “Software reliability issues under operational and testing constraints”, *Asia Pacific Journal of Operations Research.*, **22(1)**, 33-49, (2005).

[4] Cai K.-Y., Cao P., Dong Z. and Liu K. , “Mathematical modeling of software reliability testing with imperfect debugging”, *Computers & Mathematics with Applications*, **59(10)**, 3245-3285, (2010).

[5] Mane M., Joshi M., Kadam A. and Joshi S.D., “Software reliability and quality analyser with quality metric Analysis Along with software reliability growth model”, *International Journal of Computer Science and Information Technologies*, 5 (3), 3803-3806, (2014).

[6] Munson J.C., “Software faults, software failures and software reliability modeling”, *Information and Software Technology*, 38(11), 687-699, (1996).

[7] Jain M., Agrawal S.C. and Agarwal P., “Markovian software reliability model for two types of failures with imperfect debugging rate and generation of errors*”, IJE Transactions A: Basics*, **25(2)**, 177-188, (2012).

[8] Yamada S., Tokuno, K. and Osaki, S., “Software reliability measurement in imperfect debugging environment and its application”, *Reliability Engineering and System Safety*, **40**, 139-147, (1993).

[9] Yamada S., “Software reliability growth models incorporating imperfect debugging with introduced faults”, *Electronics and Communications in Japan (Part III: Fundamental Electronic Science),* **81(4)**, 33-41, (1998).

[10] Cao P., Dong Z., Liu K. and Cai K-Y, “Quantitative effects of software testing on reliability improvement in the presence of imperfect debugging”, *Information Sciences*, **218(1)**, 119-132, (2013).

[11] Peng R., Li Y. F., Zhang, Wenjuan and Hu, Q. P., “*Testing effort dependent software reliability model for imperfect debugging process considering both detection and correction”, Reliability Engineering & System Safety*, **126**, 37-43, (2014).