INFORMATION
SYSTEMS IN
MANAGEMENT

Information Systems in Management (2016) Vol. 5 (2) 175−184

MODELING THE SOFTWARE TESTING PROCESS TAKING INTO ACCOUNT THE SECONDARY ERRORS

GUSTAW KONOPACKI

Military University of Technology, Department of Cybernetics,
Institute of Information Systems

It discusses the two formal models of software testing by the concept of a black box. In the first model assumes a non-zero probability of not removing the detected error. In the second model assumes also non-zero probability to introduce additional of error, so-called secondary error. In both cases the systems of Chapman-Kolmogorov differential equations was formulated. Solving them was obtained formulas to enable an estimate the expected number of errors remaining in the software after end of testing and estimation of the expected duration of the process to complete software testing them.

Keywords: software, software testing, stochastic process

1. Introduction

The testing of a newly developed program, prior to its practical use, is a commonly followed practice. The program testing process involves the execution of the program with many sets of input data with the intention of finding errors. Testing is done to lower the chances of in-service failures which are defined as an unacceptable departure from a program operation. A long period of testing results in increasing the chances of detecting program errors and decreasing the chances of in-service failures, but it also results in increasing the cost of the program testing process.

It is known that testing is the most significant money consuming stage of the program development. The cost of the program testing process can make 50-70 percent of the total cost of the program development, especially for complex program systems [11]. Considering the essential impact of the testing cost on the whole program development cost, the testing process ought to be prudently planned and organized. Decisions relative to the testing process organization should be made on the basis of the results of testing efficiency analysis. In order to make such an analysis easier it may be convenient to estimate the number of program errors that could be encountered during the process of program testing. The knowledge of this estimation makes it possible to evaluate the duration and the cost of the program testing process, e.g. by means of formal, mathematical expressions. Such evaluations can be very useful in practice, e.g. for comparing the effectiveness of different ways of program testing process organizations (i.e. in order to find an optimal organization).

The number of program errors encountered during the testing process depends on many factors, such as the testing process organization (which defines the manner of the testing process realization), the duration of the testing, the testers' qualifications and professional experience, and the reliability level of the program at the beginning of the testing process. The duration of the program testing process can be determined by a time spent on testing activities (it may be a calendar time or so-called execution time) or by the cardinality of the set of input data used for the testing. The first way of the two mentioned above is characteristic for so-called time-domain models of software testing and the other way is specific to so-called data-domain models [7, 12].

Software testing is a finishing step process of creating systems and applications. The importance of this step underlines the fact that it plays the role of one of the key factors influencing the reliability of the software and, indirectly, the evaluation of the entire system.

Software testing process is not a single act but is an exploded during repetitive sequence of steps in which, in simple terms, to run this software on a specially prepared test data, determine whether errors in the software and delete them. The purpose of each step of testing is to detect as many errors possibly undetected in previous steps testing and removing as many of the detected errors.

On the longevity the testing process, usually expensive, affect the assessment criteria obtained level of software reliability. In general, these criteria are formulated on the basis of software reliability models (the first of which have already appeared in the 70s of the last century) and were based on an assessment of:

- the expected number of errors remaining in the software at the end of the next stage of testing,
- the expected length of the segment duration of software testing to detect another error.

Early software reliability models allow the determination of the above-mentioned size only after the test. This is a significant drawback hinders their use in planning the testing process, even in the time dimension. Therefore, there are still attempts to build such models, which would be useful for planning the software testing process. One such proposal is contained in [4] and is a generalization of previously published models by Shooman and Jelinki-Moranda.

A very strong assumption, adopted in the model [4] was the assumption that detect every error is equivalent to its removal. Since, in practice software testing error detection does not necessarily mean its immediate removal, so getting the right model [4], would require departure from the assumption. This article formulated two models that consider the possibility of not removing the detected error and, even worse, not only not removing the detected error, but the introduction of additional error during the removal of the previous one. In both models, the following assumptions about the software testing process, presented in [4]:

- before testing the software is N errors,
- errors are independent of each other, i.e. the detection and removal of any of them does not affect the detection of any of the other,
- errors are indistinguishable,
- errors are detected individually,
- at each step of the testing process starts with the simultaneous detection of all errors currently in the software,
- the testing process is a continuous process over time, and the length of time η_j elapsed from the moment next error was detected while in the software still remaining j mistakes is a random variable with exponential distribution with parameter λ_j, depending on the number of errors remaining in the software:

$$P\{\eta_j < x\} = 1 - e^{-\lambda_i x}, \quad x \geq 0, \quad j = 1, 2, ..., N, \tag{1}$$

- λ_j for all errors $j = 1, 2, ..., N$ are the same and equal λ.

In addition, pending further models (Model I and Model II) assume the following assumptions:

- Model I - error detection does not mean the absolute of his removal from of software, but it can occur with a probability $r \in [0; 1]$,
- Model II - error detection does not mean the absolute of his removal from of software, but it can occur that the with probability $q \in [0; 1]$ will be introduced an additional error, so-called. secondary error.

These models test will be used to designate the following most important characteristics of the software testing process:

- the expected value of the number of errors remaining in the software after the time t from the start of the test,
- the expected value of time from the start of testing software to stay in the moment exactly ***j*** errors (***j* = 0, 1, 2, …, *N***).

2. Model I

Adopting presented assumptions, the software testing process can be interpreted as a stochastic process **(*N*, *T*)** which denote the number of errors in the software after the time ***t*** from the start of testing this software and it is a Markov process DC class states - discrete, continuous parameter (time), where ***N*** = **{0, 1, 2, ...}** is the set of states, and ***T*** ∈ **[0; ∞)** - time.

Based on the assumptions adopted for the considered model can take the following according to the intensity of the transition between of the states of process:

$$\begin{aligned} \lambda_{j,j-1} &= jr\lambda r \\ \lambda_{j,j} &= -\,jr\lambda r \qquad j = 1, 2, ..., N\,. \end{aligned} \tag{2}$$

Graphical presentation of the transition matrix of the process is shown in Figure 1.

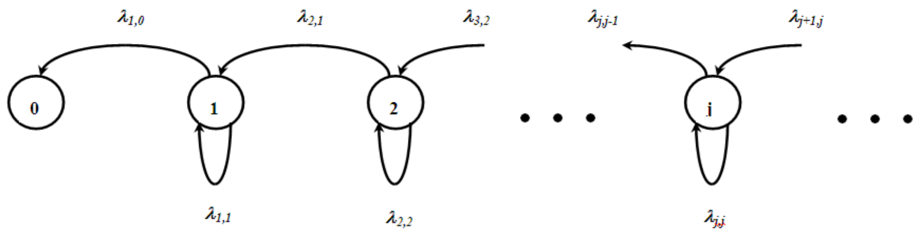

Figure1. Graphical presentation of transition matrix of a stochastic process **(*N*, *T*)** describing the Model I software testing

To determine the probability distribution vector of the process of finding the above a particular state after time t of software testing

$$p(t) = \langle p_0(t),\ p_1(t),\ p_2(t),\ ...,\ p_N(t)\rangle$$

must solve the following system of differential equations:

$$\begin{cases} p_0'(t) = (1-r)\lambda p_1(t) \\ p_1'(t) = 2(1-r)\lambda p_2(t) - (1-r)\lambda p_1(t) \\ \dots \\ p_j'(t) = j(1-r)\lambda p_{j+1}(t) - j(1-r)\lambda p_j(t), \qquad j = 2,3,\dots,N\,. \end{cases} \tag{3}$$

with the initial conditions

$$p_j(0) = \begin{cases} 0, & j \neq N, \\ 1, & j = N, \end{cases} \tag{4}$$

arising from the fact that, at the start of software testing to contain an $N > 0$ errors.

The system of equations (3) will be solved using the following generating function:

$$F(s,t) = \sum_{j=1}^{\infty} p_j(t) \cdot s^j, \qquad |s| \leq 1. \tag{5}$$

Using the function (5) the system of equations (3) and making the appropriate transformations obtained an expression for the probability that after a time t software testing remain in it the j errors:

$$p_j(t) = \begin{cases} \begin{pmatrix} N \\ j \end{pmatrix} \cdot e^{-j(1-r)\lambda t}\left(1 - e^{-(1-r)\lambda t}\right)^N, & j = 0,1,2,\dots,N \\ 0 & , \quad j > N\,. \end{cases} \tag{6}$$

Having probability (6) can determine the expected value and variance of the stochastic process in question, which describe process software testing, ie. the expected value of the number of errors remaining in the software after its testing by the time t:

$$E(N(t)) = N \cdot e^{-r\lambda t}, \qquad D^2(N(t)) = N \cdot e^{-r\lambda t}\left(1 - e^{-r\lambda t}\right). \tag{7}$$

In order to determine the expected time of the software testing until remain in the j errors $(j = 0, 1, 2, \dots, N)$, accepted the following submissions.

Let $\tau(i, j)$ is the random variable defining the residence time of the process (N, T) in a subset of states of $N_i = \{j + 1, j + 2, \dots, i\}$, i.e., the transition time of the process from the state i to the state j. Let theta $\theta(j)$ denotes a random variable defining the process residence time (N, T) in the state j.

Having regard to assumption regarding considered model of software testing as well as using the determination of the residence time homogeneous Markov process in a given set of states, you can specify the following formulas:

$$\begin{aligned} &\tau(j+1, j) = \theta(j+1), \qquad j = 0,1,2,\dots, N-1, \\ &\tau(i, j) = \theta(i) + \tau(i-1, j), \qquad i = j+2, j+3,\dots, N\,. \end{aligned} \tag{8}$$

Knowing that the distribution function of the random variable $\theta(j)$ is of the form (it follows from (1) and (2)):

$$P\{\theta(j) < x\} = 1 - e^{-jr\lambda x}, \qquad j = 1,2,\dots, N\,. \tag{9}$$

Assuming $i = N$ and solving the system of equations recursive (8) using (9), and then calculating the expected value of the $\tau(N, j)$ random variable is obtained the expected value of the time that elapses between the beginning of testing until in the software will remain j of errors will be expressed the formula

$$E(\tau(N, j)) = \frac{1}{r\lambda} \sum_{k=j+1}^{N} \frac{1}{k}, \qquad j = 0,1,2,\dots, N-1. \tag{10}$$

3. Model II

Considered further on a model of software testing - in accordance with adopted earlier assumptions - allows for a situation in which the error detected not only not will removed, but the from software will be introduced additional (secondary) error, which means that the number of errors in the software will increase by 1. Therefore, stochastic process **(*N, T*)** specifying the number errors in software after the time ***t*** from the start of testing software can be interpreted as a stochastic Markov process DC class: states discrete, continuous parameter (time).

Based on the assumptions adopted for the considered model can take the following according to the intensity of the transition between of the states of process:

$$\begin{aligned} &\mu_{j,j+1} = jq(1-r)\lambda, \\ &\lambda_{j,j} = -j(r+q-2rq)\lambda, \qquad j = 1,2,\dots \\ &\lambda_{j,j-1} = jr(1-q)\lambda \end{aligned} \tag{11}$$

Graphical presentation of the transition matrix of the process is shown in Figure 2.

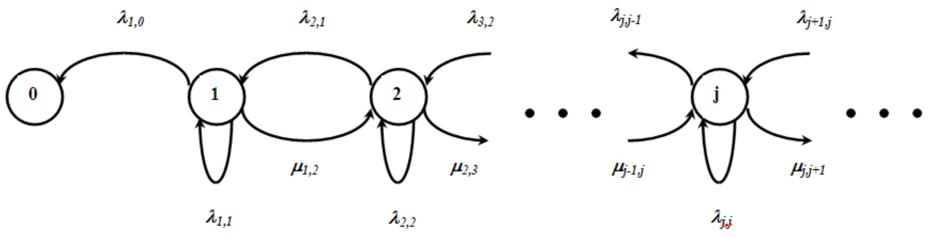

Figure 2. Graphical presentation of transition matrix of a stochastic process **(*N, T*)** describing the Model II software testing

To determine the probability distribution vector of the process of finding the above a particular state after time t of software testing

$$p(t)=\langle p_0(t), p_1(t), p_2(t), \ldots\rangle$$

must solve the following system of differential equations:

$$\begin{cases} p_0'(t)=q(1-r)\lambda p_1(t) \\ p_1'(t)=2q(1-r)\lambda p_2(t)-(r+q-2rq)\lambda p_1(t) \\ \ldots \\ p_j'(t)=(1+j)q(1-r)\lambda p_{j+1}(t)-j(r+q-2rq)\lambda p_j(t)+(j-1)r(1-q)\lambda p_{j-1}(t), \quad j=2,3,\ldots \end{cases} \tag{12}$$

with the initial conditions

$$p_j(0)=\begin{cases} 0, & j\neq N, \\ 1, & j=N, \end{cases} \tag{13}$$

arising from the fact that, at the start of software testing to contain an $N > 0$ errors.

Using the function (5) in the system of equations (12) and making the appropriate transformations obtained an expression for the expected value of the process **(*N, T*)**, i.e. the expected number of errors remaining in the software after the time t testing this software:

$$E(N,t)=N\cdot e^{(r-q)\lambda t}. \tag{14}$$

From formula (14) shows that the number of errors in the software with the passage of time his testing will be decrease when the following condition will be met:

$$r>q.$$

Let $\tau(j+i, j)$ is the random variable defining the residence time of the process **(*N*, *T*)** in a subset of states of $N_i = \{j+1, j+2, ..., j+i\}$, i.e., the time that elapses from the time when the process reached a state ***j* + *i*** until it reaches the state ***j*** for the first time. Let $\theta(j)$ means the same variable as in Model I.

In accordance with the assumptions about the model II and the definition of the residence time Markov process in the specified set of states, you can formulate the following equality:

- the case when the process **(*N*, *T*)** goes from the state ***j* + 1** to the state ***j*** or from the state ***j* + *i*** to state ***j* + *i* – 1**, which occurs with the probability

$$\frac{r(1-q)}{q(1-r)+r(1-q)},$$

  then

$$\begin{aligned}&\tau(j, j+1)=\theta(j+1), \qquad j=0,1,2,...\\&\tau(j+i, j)=\theta(j+i)+\tau(j+i-1, j), \qquad i=2,3,...\end{aligned} \tag{15}$$

- the case when the process **(*N*, *T*)** goes from the state ***j* + 1** to the state ***j* + 2** or from the state ***j* + *i*** to state ***j* + *i* + 1**, which occurs with the probability

$$\frac{q(1-r)}{q(1-r)+r(1-q)},$$

  then

$$\begin{aligned}&\tau(j+1, j)=\theta(j+1)+\tau(j+2, j), \qquad j=0,1,2,...\\&\tau(j+i, j)=\theta(j+i)+\tau(j+i+1, j), \qquad i=2,3,...\end{aligned} \tag{16}$$

Determination of the expected value of a random variable $\tau(j+i, j)$ is possible by solving the following system of algebraic equations:

$$\begin{cases}E(\tau(j+1,j))=E(\theta(j+1))+\dfrac{q(1-r)}{q(1-r)+r(1-q)}\cdot E(\tau(j+2,j))\\E(\tau(j+2,j))=E(\theta(j+2))+\dfrac{q(1-r)}{q(1-r)+r(1-q)}\cdot E(\tau(j+3,j))+\dfrac{r(1-q)}{q(1-r)+r(1-q)}\cdot E(\tau(j+1,j))\\...\\E(\tau(j+i,j))=E(\theta(j+i))+\dfrac{q(1-r)}{q(1-r)+r(1-q)}\cdot E(\tau(j+i+1,j))+\dfrac{r(1-q)}{q(1-r)+r(1-q)}\cdot E(\tau(j+i-1,j))\\\qquad i=3,4,...\end{cases} \tag{17}$$

Since the system of equations (17) is unlimited, for its solution cannot be applied classical methods of solving algebraic equations limited. In [3] proposed an iterative method for solving of system of equations (17), permitting to obtain a satisfactory estimate of the expected value of the random variable ***τ(j + i, j)***,which is expressed by the following formula:

$$E(\tau(j+i,j)) \le \frac{1}{(r-q)\lambda}\sum_{k=0}^{i}\frac{1}{j+i}, \qquad j=0,1,2,..., \qquad i=j+1, j+2,... \tag{18}$$

Hence, an estimate of the expected value of the time that has elapsed since the start of testing until the software is error j expresses the relation:

$$E(\tau(N,j)) = \frac{1}{(r-q)\lambda}\sum_{k=j+1}^{N}\frac{1}{k}, \quad j=0,1,2,...,N-1. \tag{19}$$

4. Conclusions

Formula obtained from the analysis of the presented models of software testing are consistent with intuition: the higher the intensity λ error detection, and the greater the probability ***r*** to remove the detected error, the shorter the duration of software testing. The inclusion in the Model II, the probability ***q*** introduction of a secondary error getting the right results achieved and allows you to use them in the practice of software testing.

Adopted in the article, the parameters λ, ***r***, ***q***, and ***N*** are characterized some aspects of both the process of designing, manufacturing and testing software. The initial number ***N*** of errors in the software will depend crucially on the method and tools for designing and producing the well-known software complexity. The λ intensity of the error detection is derived from the method used for software testing, while the probability ***r*** and ***q*** characterize the skills and experience of the team of testers.

Presented the formulas (7), (10), (14) and (19) were obtained on the assumption that the parameters λ, ***r***, ***q***, and ***N*** are fixed, although in practice they are not usually precisely known and can only be estimated (e.g. [4]), but it does not call into question the obtained results. In the case where there is a possibility to determine the probability distribution of the number N of errors in the software prior to its test as a discrete random variable, as listed formulas has adopted the form:

$$E\left(N(t)\right)=\sum_{n=0}^{\infty} n\cdot e^{-r\lambda t}\cdot P(N=n),$$
$$E\left(\tau(N,j)\right)=\frac{1}{r\lambda}\sum_{n=0}^{\infty}\sum_{k=j+1}^{n}\frac{1}{k}\cdot P(N=n),\quad j=0,1,2,\ldots,N-1.,$$
$$E(N,t)=\sum_{n=0}^{\infty} n\cdot e^{(r-q)\lambda t}\cdot P(N=n),\tag{20}$$
$$E\left(\tau(N,j)\right)=\frac{1}{(r-q)\lambda}\sum_{n=0}^{\infty}\sum_{k=j+1}^{n}\frac{1}{k}\cdot P(N=n),\quad j=0,1,2,\ldots,N-1.$$

In the process of software testing, with the passage of time, all the size change: decrease $\boldsymbol{N}$, $\boldsymbol{\lambda}$ and $\boldsymbol{q}$, and $\boldsymbol{r}$ is increasing, but in this article includes only a change in the number of errors in the software. Of course, a similar approach as above can be used when is possible to estimate distributions of continuous random variables $\boldsymbol{\lambda}$, $\boldsymbol{r}$, $\boldsymbol{q}$.

REFERENCES

[1] Feler W. (1996) *Wstęp do rachunku prawdopodobieństwa*, PWN, Warszawa.

[2] Haggstrom O. (2001) *Finite markov chains and algorithmic applications*, Chalmers University of Technology.

[3] Konopacki G., Pluciński I. (1989) *O pewnych modelach testowania oprogramowania*, in: Biuletyn WAT w Warszawie, Nr 4/1989, s. 13-23.

[4] Konopacki G., Worwa K. (1984) *Uogólnienie modeli niezawodności oprogramowania Shoomana i Jelinskiego-Morandy*, in: Biuletyn WAT w Warszawie, Nr 12/1984, s. 105-109.

[5] Lawler G.F. (1995*) Introduction to Stochastic Processes*, Chapman & Hall / CRC.

[6] Mitzenmacher M., Upfal E. (2009) *Metody probabilistyczne i obliczenia*, WNT, Warszawa.

[7] Musa J.D., Iannino A., Okumoto K. (1987) *Software reliability. Measurement, prediction, application*. McGraw-Hill, Inc.

[8] Norris J. R. (1977) *Markov Chains*, Cambridge Series in Statistical and Probabilistic Mathematics.

[9] Papoulis A. (1972) *Prawdopodobieństwo, zmienne losowe i procesy stochastyczne*, WNT, Warszawa.

[10] Ross S.M. (1996) *Stochastic processes*. John Wiley & Sons, New York.

[11] Thayer T.A., Lipov M., Nelson E.C. (1978) *Software reliability*. North-Holland Publishing Company. Amsterdam.

[12] Zahedi F., Ashrafi N. (1991) *Software reliability based on structure, utility, price and cost*. IEEE Trans. on Software Engrg. Vol.17, No. 4, 345–356.