MODELLING THE SOFTWARE TESTING PROCESS

Kazimierz Worwa

Faculty of Cybernetics, Military University of Technology, 00-908 Warszawa, Kaliskiego 2, Poland kworwa@wat.edu.pl

Abstract. An approach to formal modelling the program testing process is proposed in the paper. Considerations are based on some program reliability-growth model that is constructed for assumed scheme of the program testing process. In this model the program under the testing is characterized by means of so-called characteristic matrix and the program testing process is determined by means of so-called testing strategy. The formula for determining the mean value of the predicted number of errors encountered during the program testing is obtained. This formula can be used if the characteristic matrix and the testing strategy are known. Formulae for evaluating this value when the program characteristic matrix is not known are also proposed in the paper.

Keywords: software testing, software reliability, software reliability models.

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. A cost of the program testing process can make even more than 50% of the total cost of the program development, especially for complex program systems [12, 17]. 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 results of testing efficiency analysis. In order to make such analysis easier it may be convenient to evaluate the number of program errors that could be encountered during the program testing process. The knowledge of this evaluation results makes it possible to evaluate the duration and the cost of the program testing process, e.g. by means of formal, mathematical expressions. Such estimations 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).

For the purpose of the rational prediction of the program testing process duration software reliability-growth models are recommended in this paper. These models can provide a first program reliability-estimate and support project management in planning further development processes. Therefore many software reliability-growth models have been proposed in the literature [1–5, 7–9, 11, 13–14, 18, 20, 24–26]. The number of models that have actually been employed to assist with software development projects is, however, much smaller. The reason for this is that none of these models give sufficiently accurate estimates of reliability. This seems to be chiefly due to the fact that the authors of the various models have paid little or no attention to the manner in which a software is tested. Software reliability modelling has become one of the most important aspects in software reliability engineering since Jelinski-Moranda [10] and Schooman [16] models appeared. Various methodologies have been adopted to model software reliability behaviour. The most of existing work on software reliability modelling is focused on continuous-time base, which assumes that software reliability behaviour can be measured in terms of time. It may be a calendar time, a clock time or a CPU execution time [1, 3, 7–9, 11, 13–14, 17, 20, 24, 26]. Although this assumption is appropriate for a wide scope of software systems, there are many systems, which are essentially different from this assumption. For example, reliability behaviour of a reservation software system should be measured in terms of how many reservations are successful, rather than how long the software operates without any failure. Similarly, reliability behaviour of a bank transaction processing software system should be assessed in terms of how many transactions are successful, etc. Obviously, for these systems, the time base of reliability measurement is essentially discrete rather than continuous. Models that are based on a discrete-time approach are called input-domain or run-domain models [2, 4–5, 22–23, 25]. They usually express reliability as the probability that an execution of the software is successful. The paper [2] has been proposed the terms a run and run reliability and a conceptual framework of software run reliability modelling. A run is a minimum execution unit of software and run reliability means the probability that software successfully perform a run. The concrete sense of a run is subject to application context. For example, a run can correspond to execution of a test case, of a program path, etc.

This paper attempts to describe a new discrete-time software reliability-growth model for some scheme of the program testing and to determine a formula to evaluate the predicted number of program errors encountered during the testing process.

2. The program testing scheme

An organization of a program testing process depends on the program testing strategy which was selected for the testing. In particular, this strategy defines the way of the testing process realization and the set of program input data which is used for the testing. It is assumed that the program testing process consists of a number of organizational units of the program testing phase that are called the testing stages. Every stage of the program testing process consists of the two following steps of testing:

— testing of the program under consideration with a prepared set of program input data (tests),
— comparison of the results with the expected outputs for the data used and removing all errors encountered during testing.

It is noteworthy that all program faults discovered during the first step of some testing stage are only registered and removed only this step is finished. Such organization of the testing stage means that testing and debugging are performed in different steps (not simultaneously) and consequently some program fault can be observed more than once in the testing stage. The paper [25] has been underlined that above definition of the testing scheme is used in the most mathematical models of software testing.

Let S mean the program testing strategy which is defined as follows

$$S = (K, (L_1, L_2, \ldots, L_k, \ldots, L_K)) \tag{1}$$

where: K – the number of stages of the program testing process,
L_k – the number of program input data used in the k-th program testing stage, $L_k > 0$, $k = \overline{1, K}$

Execution of the program under the testing process with one input data set (test case) will be called a run in this paper. The run can be successful, if program execution did not lead to encounter any program errors or not successful, if program execution was incorrect, i.e. some errors were encountered.

Let S denote the set of all strategies that have the form (1). The strategy (1) defines a program testing scheme. In accordance with this scheme, the process of removing program errors which were encountered during the k-th program testing stage can be started after the execution of the program on all L_k tests is finished.

According to assumed testing scheme a situation that a number of different tests of all L_k tests executed during the k-th stage encounter the same error in the program under the testing is possible. So, according to the note mentioned above, a situation that several runs will lead to encounter the same program error is possible.

Let p_{nm} define the probability of an event that n errors will be encountered during a single stage of the program testing if there are m tests that have incorrect execution in that stage. If we assume that every execution of the program with a single test can lead to encounter at most one program error, we will have

$$0 \leq p_{nm} \leq 1 \quad \text{if } n \leq m,\ n \geq 0 \tag{2}$$

where in particular

$$\begin{aligned} &p_{00} = p_{11} = 1 \\ &p_{nm} = 0 \quad \text{if } n > m \end{aligned} \tag{3}$$

and

$$\sum_{n=0}^{\infty} p_{nm} = 1, \quad m \in \{0,1,2,\ldots\}. \tag{4}$$

Probabilities p_{nm}, $n \in \{0,1,2,\ldots,m\}$, $m \in \{0,1,2,\ldots\}$, form an infinite matrix $P = [p_{nm}]$ as follows

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 1 & p_{12} & p_{13} & p_{14} & \cdots \\ 0 & 0 & p_{22} & p_{23} & p_{24} & \cdots \\ 0 & 0 & 0 & p_{33} & p_{34} & \cdots \\ 0 & 0 & 0 & 0 & p_{44} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \tag{5}$$

where values p_{nm} are defined by (2)–(4).

The matrix P contains the values 0 below the main diagonal because – in accordance with earlier assumption that every execution of the program with a single test can lead to encounter at most one program error – it is not possible to encounter more different errors than the number of incorrect runs.

The values of probabilities p_{nm} for every program testing stage only depend on the number of tests which are used during that stage.

The values of probabilities p_{nm} that form the matrix P depend on a logical structure of the program. In particular, an important impact on these probabilities has:

— number of paths that have been identified in the program,
— degree of covering individual paths, that can be measured by number of program instructions that belong to two or more paths,
— length of individual paths, measured by number of program instructions that are executed in case of path activation.

The matrix P will be called the characteristic matrix of the program under testing.

3. Program reliability coefficient

Let $M_k(S, P)$ denote the number of runs which lead to incorrect execution of the program under the testing, i.e. to encounter program errors, during the k-th stage of program testing process according to the testing strategy S and the characteristic matrix P.

Let $N_k(S, P)$ mean the total number of errors encountered during the k-th stage of program testing process in accordance with the testing strategy S and the characteristic matrix P.

While planning the program testing process it is reasonable to treat the values $M_k(S, P)$ and $N_k(S, P)$, $k = \overline{1, K}$, as random variables, $N_k(S, P) \leq M_k(S, P)$, $k = \overline{1, K}$.

Joint distribution of the random variables $N_k(S, P)$, $M_k(S, P)$ can be determined as follows:

$$\begin{aligned} &Pr\{N_k(S, P) = n_k, M_k(S, P) = m_k\} \\ &\quad = Pr\{N_k(S, P) = n_k / M_k(S, P) = m_k\} Pr\{M_k(S, P) = m_k\} \end{aligned} \tag{6}$$

Probability distribution of the random variable $N_k(S, P)$ can be determined as a marginal distribution in a distribution of two-dimensional random variable $(N_k(S, P), M_k(S, P))$:

$$\begin{aligned} &Pr\{N_k(S, P) = n_k\} \\ &= \sum_{m_k=0}^{L_k} Pr\{N_k(S, P) = n_k / M_k(S, P) = m_k\} Pr\{M_k(S, P) = m_k\} \end{aligned} \tag{7}$$

Let $N(S, P)$ mean the total number of errors encountered during the process of the program testing in accordance with the testing strategy S and the characteristic matrix P. The value $N(S, P)$ is a random variable and can be determined as follows:

$$N(S, P) = \sum_{k=1}^{K} N_k(S, P). \tag{8}$$

Taking into account assumed testing scheme probability distribution of the random variable N(S,P) has the form:

$$
\begin{aligned}
&Pr\{N(S,P)=n_k\}\\
&=\sum_{n_1+n_2+\ldots+n_k=n} Pr\{N_1(S,P)=n_1,N_2(S,P)=n_2,\ldots,N_K(S,P)=n_K\},\\
&\qquad n=0,1,2,\ldots
\end{aligned}
\tag{9}
$$

where probabilities $Pr\{N_1(S,P)=n_1,N_2(S,P)=n_2,\ldots,N_K(S,P)=n_K\}$ determine joint distribution of K-dimentional random variable $(N_1(S,P),N_2(S,P),\ldots,N_K(S,P))$.

Probability distribution $Pr\{N_1(S,P)=n_1,N_2(S,P)=n_2,\ldots,N_K(S,P)=n_K\}$ can be determined as follows [21]

$$
\begin{aligned}
&Pr\{N_1(S,P)=n_1,N_2(S,P)=n_2,\ldots,N_K(S,P)=n_K\}\\
&=\prod_{k=1}^{K}\sum_{m_k=0}^{L_k} Pr\{N_k(S,P)=n_k/M_k(S,P)=m_k\}\\
&\quad\cdot Pr\{M_k(S,P)=n_k/N_i(S,P)=n_i,\ i=\overline{1,k-1}\}
\end{aligned}
\tag{10}
$$

where

$$
\begin{aligned}
&Pr\{M_k(S,P)=m_k/N_i(S,P)=n_i,\ i=\overline{1,k-1}\}\\
&=Pr\{M_k(S,P)=m_k/N_1(S,P)=n_1,N_2(S,P)=n_2,\\
&\quad\ldots,N_{k-1}(S,P)=n_{k-1}
\end{aligned}
\tag{11}
$$

and $N_0(S,P)=0$.

According to earlier denotations we have

$$
p_{nm}=Pr\{N_k(S)=n|_{M_k(S)=m}\},\quad k=\overline{1,K-1},
\tag{12}
$$

so formula (10) takes a form:

$$
\begin{aligned}
&Pr\{N_1(S,P)=n_1,N_2(S,P)=n_2,\ldots,N_K(S,P)=n_K\}\\
&=\prod_{k=1}^{K}\sum_{m_k=0}^{L_k} p_{n_k m_k} Pr\{M_k(S,P)=m_k/N_i(S,P)=n_i,\ i=\overline{1,k-1}\}
\end{aligned}
\tag{13}
$$

By the substitution of this equation into (9), we get:

$$\begin{aligned} Pr\{N(S,P)=n\} &= \sum_{n_1+n_2+\ldots+n_K=n} Pr\{N_1(S,P)=n_1, N_2(S,P)=n_2 \\ &\quad ,\ldots, N_K(S,P)=n_K\} \\ &= \sum_{n_1+n_2+\ldots+n_K=n} \prod_{k=1}^{K} \sum_{m_k=0}^{L_k} p_{n_k m_k} Pr\{M_k(S,P)=m_k / N_i(S,P)=n_i, \\ &\quad i=\overline{1,k-1}\} \end{aligned} \tag{14}$$

As a result of program testing process execution, in accordance with the assumed testing strategy S, $N(S,P)$ program errors will be encountered. Successful removing these errors will increase the level of program reliability. The number of encountered errors $N(S,P)$ depends on both assumed testing strategy and initial level of program reliability, i.e. reliability of the program at the beginning of the program testing process.

It is very well known that in spite of successful testing phase realization the program can still contain some errors. Bearing this fact in mind it is rationally to describe a result $X(S,P)$ of one program execution, after finishing the testing process according to strategy S, as follows:

$$X(S,P) = \begin{cases} 1 & \text{if an execution of the program with characteristic} \\ & \text{matrix } P\text{,that was tested in accordance with the} \\ & \text{strategy } S\text{, is correct;} \\ 0 & \text{otherwise.} \end{cases}$$

Planning the phase of program testing process it is rationally to treat amounts $N(S,P)$ and $X(S,P)$ as dependent random variables, because the result $X(S,P)$ of program execution for some input data set, after finishing the testing according to strategy S, depends on a number of errors $N(S,P)$ encountered during this process.

Let $r(S,P)$ mean a probability of a correct execution of the program with the characteristic matrix P, after finishing the testing process according to the strategy S, i.e.

$$r(S,P) = Pr\{X(S,P)=1\}. \tag{15}$$

The probability $r(S,P)$ will be treated as a program reliability measure in the paper.

We can write:

$$r(S,P)=\sum_{n=0}^{L(S)} r(S,P)|_{N(S,P)=n}\cdot Pr\{N(S,P)=n\}, \tag{16}$$

where: $r(S,P)|_{N(S,P)=n}$ – conditional probability of a correct execution of the program with the characteristic matrix P if the testing process, according to the strategy S, leaded to encounter $N(S,P)=n$ program errors; $L(S)$ – a total number of runs that are performed during testing the program according to the strategy S, i.e.

$$L(S)=\sum_{k=1}^{K} L_k. \tag{17}$$

By substitution of expression (14) into (16), we get

$$\begin{aligned} r(S,P) &= \sum_{n=0}^{L(S)} r(S,P)|_{N(S,P)=n} \\ &= \sum_{n_1+n_2+\ldots+n_K=n} \prod_{k=1}^{K} \sum_{m_k=0}^{L_k} p_{n_k m_k} Pr\{M_k(S,P)=m_k/N_i(S,P)=n_i \\ &\quad i=\overline{1,k-1}\}. \end{aligned} \tag{18}$$

If a number of errors is encountered in the program and they are successfully corrected, the program reliability level will increase. We will describe it as follows:

$$r(S,P)|_{N(S,P)=n}=r+\Delta r(P,n), \tag{19}$$

where: r – an initial value of the program reliability coefficient, i.e. value of the program reliability coefficient at the beginning of the program testing process,
$\Delta r(P,n)$ – an increase of the reliability coefficient value of the program with the characteristic matrix P, as a result of encountering and removing n errors.

On the basis of facts described in literature (see e.g. [13, 17, 19]) it is assumed that the increase $\Delta r(P,n)$ is of the form:

$$\Delta r(P,n)=(1-r)(1-e^{\alpha n}), \tag{20}$$

where α is a parameter that characterizes both the internal structure of the program under testing and the impact of one error removal on the increase of the program reliability.

It should be stated that evaluating values of both parameters r and α relies on the testing history of the software in practice.

Thus, by substituting the last equation into (19), we get:

$$r(S,P)|_{N(S,P)=n} = 1 - (1-r)e^{-\alpha m}. \tag{21}$$

It is easy to notice that assumed testing scheme, such that probability of successful performance of one run remains constant during testing stage, leads to the so-called Bernoulli scheme. Consequently, conditional probabilities from (18) can be determined by means the binominal distribution as follows:

$$\begin{aligned} &Pr\{M_k(S,P) = m_k/N_i(S,P) = n_i,\ i = \overline{1,k-1}\} \\ &= \binom{L_k}{m_k} [e^{-\alpha \sum_{i=1}^{k=1} n_i}(1-r)]^{m_k} [1 - e^{-\alpha \sum_{i=1}^{k=1} n_i}(1-r)]^{L_k - m_k}, \\ &m_k \in \{0,1,2,\cdots,L_k\},\ k = \overline{1,K}. \end{aligned} \tag{22}$$

Substituting (21) and (22) into (18), we get the following expression for the program reliability coefficient $r(S,P)$, after finishing the testing process according to the strategy S:

$$r(S,P) = 1 - (1-r)A(S,P), \tag{23}$$

where

$$A(S,P) = \sum_{n_1=0}^{L_1} \sum_{n_2=0}^{L_2} \cdots \sum_{n_K=0}^{L_K} e^{-\alpha \sum_{k=1}^{K} n_k} \prod_{k=1}^{K} \sum_{m_k=0}^{L_k} p_{n_k m_k} A_{m_k}\left(\sum_{i=1}^{k-1} n_i, L_k\right) \tag{24}$$

and

$$\begin{aligned} &A_{m_k}\left(\sum_{i=1}^{k-1} n_i, L_k\right) \\ &= \binom{L_k}{m_k} [e^{-\alpha \sum_{i=1}^{k-1} n_i}(1-r)]^{m_k} [1 - e^{-\alpha \sum_{i=1}^{k-1} n_i}(1-r)]^{L_k - m_k}, \\ &m_k \in \{0,1,2,\cdots L_k\},\ k = \overline{1,K}. \end{aligned} \tag{25}$$

The quantity $A_{m_k}\big(\sum_{i=1}^{k-1} n_i, L_k\big)$, $m_k \in \{0,1,2,\cdots,L_k\}$, $n_i \in \{0,1,2,\cdots,L_k\}$, $k \in \{1,2,\cdots,K\}$, means probability that m_k runs of all L_k runs performed during the k-th testing stage will be incorrect, i.e. will lead to encounter errors on condition that in the previous $k-1$ testing stages $\sum_{i=1}^{k-1} n_i$ errors were encountered and removed.
It is easy to check that

$$\sum_{m_k=0}^{L_k} A_{m_k}\left(\sum_{i=1}^{k-1} n_i, L_k\right) = [1-e^{-\alpha\sum_{i=1}^{k-1} n_i}(1-r)+e^{-\alpha\sum_{i=1}^{k-1} n_i}(1-r)]^{L_k} = 1$$

and

$$\sum_{n_k=0}^{L_k}\sum_{m_k=0}^{L_k} p_{n_k m_k} A_{m_k}\left(\sum_{i=1}^{k-1} n_i L_k\right) = \sum_{m_k=0}^{L_k} A_{m_k}\left(\sum_{i=1}^{k-1} n_i, L_k\right)\sum_{n_k=0}^{L_k} p_{n_k m_k}$$
$$= \sum_{m_k=0}^{L_k} A_{m_k}\left(\sum_{i=1}^{k-1} n_i, L_k\right) = 1.$$

The increase of the program reliability coefficient $\Delta r(S,P) = r(S,P) - r$, as a result of testing phase realization according to the strategy S, is of the form:

$$\Delta r(S,P) = (1-r)(1-A(S,P)), \tag{26}$$

where the quantity $A(S,P)$ is determined by (24).
Let $S', S'' \in \boldsymbol{S}$ denote the following program testing strategies:

$$S' = (L, \underbrace{(l, l, \cdots, l)}_{L \text{ times}}), \tag{27}$$

$$S'' = (1, L). \tag{28}$$

It could be said that the above strategies S' and S'' are extreme forms of the testing strategy (1). In particular, if the testing strategy has the form S' than it means that in every testing stage the program under testing process is executed only once. Therefore, if the program execution encounters any fault it will be removed immediately, i.e. before next testing stage will be started. For the strategy S' becomes impossible to observe repeated appearances of faults in any testing stage. Thus this strategy is very profitable from point of view of potential increase the program reliability, but rather inefficient from economical aspect of the program testing process.

If the testing strategy has the form S'' than it means that program testing phase consists of only one testing stage containing all tests. Obviously, this strategy is very attractive because of low cost of program testing process, but probably it would not guarantee the sufficient increase of the program reliability.
If the strategies S' or S'' are used the formulae for program reliability coefficient $r(S,P)$ will take the simplified forms:

$$r(S',P)r(S') = 1-(1-)\sum_{n_1=0}^{1} e^{-\alpha n_1}(1-r)^{n_1}r^{1-n_1}$$
$$\sum_{n_2=0}^{1} e^{-\alpha n_2}[e^{-\alpha n_1}(1-r)]^{n_2}[1-e^{-\alpha n_1}(1-r)]^{1-n_2}$$
$$\dots\sum_{n_K=0}^{1} e^{-\alpha n_K}[e^{-\alpha\sum_{m=1}^{K-1}n_m}(1-r)]^{n_K}[1-e^{-\alpha\sum_{m=1}^{K-1}n_m}(1-r)]^{1-n_k} \tag{29}$$

and

$$r(S'',P) = \sum_{n=0}^{L} e^{-\alpha n}\sum_{m=0}^{L} p_{nm}\binom{L}{m}(1-r)^m r^{L-m}. \tag{30}$$

The main advantage of using the strategy S' consists in avoiding the possibility of encountering the same program error by different tests during individual testing stages. From one hand, it makes possible to increase in the effectiveness of the testing process that could me measured, for example, by a total number of encountered fault with relation to total number of tests, which were used in the testing process. From the other hand, the main disadvantage of this strategy is connected with its economic ineffectiveness because of both high cost and long duration of the testing process. Practical application of the strategy S'' can be connected with the effect of recapturing the same program errors by different tests during individual testing stages. Obviously, this effect can meaningfully decrease in the effectiveness of the testing process. In particular, if the program under the testing process has an incorrect instruction at the beginning of its source code then it is possible that all tests used in some testing stage will encounter the same program error, related to this incorrect instruction. Because of mentioned reasons, every practical testing strategy has character of some compromise between the strategies S' and S''.
Let $\boldsymbol{P}$ mean the set of all matrices that have form P, i.e:

$$\boldsymbol{P} = \{P=[p_{nm}],\ n,m\in\{0,1,2,\dots\}:\ \text{probabilities } p_{nm}$$
$$\text{meet constraints defined by } (2)-(4)\},$$

where every individual program is characterized only by one characteristic matrix $P \in \boldsymbol{P}$.
Let P^*, P^{**} denote characteristic matrices of the program under the testing that have form:

$$P^* = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots \\ 0 & 1 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & \dots \\ 0 & 0 & 0 & 1 & \dots \\ 0 & 0 & 0 & 1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \quad P^{**} = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots \\ 0 & 1 & 1 & 1 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{31}$$

It can be proved [23] that if $P^*, P^{**} \in \boldsymbol{P}$ are the matrices of the form (31) then

$$\Delta r(S, P^*) = \max_{P \in \boldsymbol{P}} \Delta r(S, P) \tag{32}$$

and

$$\Delta r(S, P^{**}) = \min_{P \in \boldsymbol{P}} \Delta r(S, P). \tag{33}$$

The dependencies (32) and (33) make possible to obtain the following evaluation of the program reliability coefficient value for any program testing strategy $S \in \boldsymbol{S}$:

$$r(S, P^{**}) \leq r(S, P) \leq r(S, P^*). \tag{34}$$

Double inequality (34) is a direct conclusion from (32) and (33). This inequality enables to two-side rough estimate of the program reliability coefficient value after the finishing the program testing process, according to the testing strategy S. This estimate is better, i.e. more precise, than $0 \leq r(S, P) \leq 1$ and may be useful in situation when the probabilities p_{nm}, that define the program characteristic matrix P, can not be determined in practice.

4. Evaluating the mean value of the number of errors encountered during the program testing process

Substituting (22) into (14), we get the following expression for probability distribution of the total number of errors encountered during the program testing process, in accordance with the testing strategy S and the characteristic matrix P:

$$\begin{aligned} & Pr\{N(S,P)=n\} \\ & = \sum_{n_1+n_2+\cdots+n_K=n} Pr\{N_1(S)=n_1, N_2(S)=n_2, \cdots, N_k(S)=n_K\} \\ & = \sum_{n_1+n_2+\cdots+n_K=n} \prod_{l=1}^{K} \sum_{m_l=0}^{L_l} p_{n_1 m_1} A_{m_1}\Big(\sum_{i=1}^{l-1} n_i, L_1\Big), \quad n=\overline{0, L(S)} \end{aligned} \tag{35}$$

where the quantities $A_{m_1}\big(\sum_{i=1}^{l-1} n_i, L_1\big)$ and $L(S)$ are determined by (25) and (17), respectively.

The knowledge of the probability distribution function (35) makes possible to obtain a formula for the mean value of the number of errors encountered during the program testing process according to the strategy S. We have

$$E[N(S,P)] = \sum_{n=0}^{L(S)} nPr\{N(S,P)=n\}, \tag{36}$$

and then, according to (35):

$$\begin{aligned} E[N(S,P)] &= \sum_{n=0}^{L(S)} n \sum_{n_1+n_2+\cdots+n_K=n} \prod_{l=1}^{K} \sum_{m_1=0}^{L_1} p_{n_1 m_1} A_{m_1}\Big(\sum_{i=1}^{l-1} n_i, L_1\Big) \\ &= \sum_{k=1}^{K} \sum_{n_1=0}^{L_1} \sum_{n_2=0}^{L_2} \dots \sum_{n_K=0}^{L_K} n_k \prod_{l=1}^{K} \sum_{m_1=0}^{n_1} p_{n_1 m_1} A_{m_1}\Big(\sum_{i=1}^{l-1} n_i, L_1\Big). \end{aligned} \tag{37}$$

The formula (37) will be simplified if the program characteristic matrix P is of the form (31), i.e.:

$$\begin{aligned} & E[N(S,P^*)] \\ & = (1-r)\sum_{k=1}^{K} L_k \sum_{n_1=0}^{L_1} \sum_{n_2=0}^{L_2} \dots \sum_{n_{k-1}=0}^{L_{k-1}} e^{-\alpha \sum_{i=1}^{k-1} n_i} \prod_{l=1}^{k-1} A^*_{n_1}\Big(\sum_{i=1}^{l-1} n_i, L_1\Big), \end{aligned} \tag{38}$$

and

$$\begin{aligned} & E[N(S,P^{**})] \\ & = \sum_{k=1}^{K} \sum_{n_1=0}^{1} A^{**}_{n_1}(0, L_1) \dots \sum_{n_{k-1}=0}^{1} A^{**}_{n_{k-1}}\Big(\sum_{i=1}^{k-2} n_i, L_{k-1}\Big) A^{**}_1\Big(\sum_{i=1}^{k-1} n_i, L_k\Big), \end{aligned} \tag{39}$$

where

$$A^*_{n_1}\left(\sum_{i=1}^{l-1} n_i, L_1\right) = [e^{-\alpha \sum_{i=1}^{l-1} n_i}(1-r)]^{n_1}[1-e^{-\alpha \sum_{i=1}^{l-1} n_i}(1-r)]^{L_1-n_1} \tag{40}$$

and

$$A^{**}_{n_k}\left(\sum_{i=1}^{k-1} n_i, L_k\right) = \{1-[1-e^{-\alpha \sum_{i=1}^{k-1} n_i}(1-r)]^{L_k}\}^{n_k}\{1-e^{-\alpha \sum_{i-1}^{k-1} n_i}(1-r)\}^{L_k(1-n_k)}, \tag{41}$$

For example, if $P = P^*$ and $S = (2, (1, 1))$, we will have

$$E[N(S, P^*)] = (1-r)[1+r+e^{-\alpha}(1-r)].$$

In practice, the formula (37) for evaluating the mean value of the number of errors encountered during the program testing process can be used if the program characteristic matrix P is known. If the probabilities $p_{n_k m_k}$, $n_k m_k \in \{0, 1, 2, \ldots, L_k\}$, $k = \overline{1, K}$, are unknown, it is possible to determine the boundary values of this evaluation.
Let P^*, P^{**} denote characteristic matrices of the program under the testing of forms (31). Then, as proved in [23], for both any program testing strategy S and any characteristic matrix P is:

$$E[N(S, P^{**})] \leq E[N(S, P)] \leq E[N(S, P^*)], \tag{42}$$

where the quantities $E[N(S, P^*)]$ and $E[N(S, P^{**})]$ are determined by (38) and (39), respectively.

5. Conclusions

In order to apply the methodology proposed in this paper, software run reliability data must be available. Compared to the amount of software reliability data reported in the literature, the amount of discrete-time software reliability data is rather limited in relation to continuous-time software reliability data, although some authors have published their own data.
It is noteworthy that the parameters (including r, α and probabilities that form the characteristic matrix P) of the presented in the paper program

reliability-growth model can be evaluated by using a process known as life testing or statistical-usage testing, in which long-term behaviour of the software is observed and values of these parameters are estimated on the basis of the observations (see e.g. [6, 17]).

Sensible estimation of the values such parameters as r, α and p_{nm} requires a suitable data base of facts that contains information about similar former programs under testing. As far as the data base of facts is concerned, it is an important component of the software engineering. data base of facts maintenance is especially useful if a software development process is stable in a domain sense. The fundamental information that creates this data base of facts concerns the backgrounds, the circumstances, the reasons and the time of the program errors encounter. Detailed analysis of data included in the data base of facts enables better understanding mutual relations between the types of software and the amount of program errors encountered during development process. Such mathematical and statistical methods as regression analysis, variation analysis, least square method, maximum likelihood method etc. are the most popular for estimating the values of parameters that are commonly used in software reliability models. The practical usefulness of the parameter estimation methods mentioned above can be confirmed by the results of researches described in several papers (see e.g. [6, 13, 17]).

References

[1] S. Basu, N. Ebrahimi, *Bayesian software reliability models based on martingale processes*, Technometrics, Vol. 45, pp. 150–158, 2003.

[2] K.Y. Cai, *Towards a conceptual framework of software run reliability modeling*, Information Science, Vol. 126, No. 6, pp. 137–163, 2000.

[3] M. Chen, A.P. Mathur, V. Rego, *Effect of testing techniques on software reliability estimates obtained using a time-domain model*, IEEE Transactions on Reliability, Vol. 44, No. 1, pp. 97–103, 1995.

[4] T.Y. Chen, Y.T. Yu, *On the relationship between partition and random testing*, IEEE Transactions on Software Engineering, Vol. 20, No. 12, pp. 977–980, 1994.

[5] T.Y. Chen, Y.T. Yu, *On the expected number of failures detected by subdomain testing and random testing*, IEEE Transactions on Software Engineering, Vol. 22, No. 2, pp. 109–119, 1996.

[6] R.H. Cobb, H.D. Mills, *Engineering software under statistical quality control*, IEEE Software, Vol. 16, pp. 44–54, 1990.

[7] A. Csenki, *Bayes predictive analysis of a fundamental software reliability model*, IEEE Transactions on Software Engineering, Vol. 39, No. 2, pp. 177–183, 1990.

[8] O. Gaudoin, *Software reliability models with two debugging rates*, International Journal of Reliability, Vol. 6, No. 1, pp. 31–42, 1999.

[9] Y. Hayakawa, G. Telfar, *Mixed poisson-type processes with application in software reliability*, Mathematical and Computer Modelling, Vol. 31, pp. 151–156, 2000.

[10] Z. Jelinski, P.B. Moranda, *Software Reliability Research*, Statistical Computer Performance Evaluation, Academic Press, New York, 1972.

[11] D.R. Jeske, H. Pham, *On the maximum likelihood estimates for the Goel-Okumoto software reliability model*, The American Statistician, Vol. 3, pp. 219–222, 2001.

[12] E. Kit, *Software testing in the real world*, ACM Press Books, 1995.

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

[14] K. Sawada, H. Sandoh, *Continuous model for software reliability demonstration testing considering damage size of software failures*, Mathematical and Computer Modelling, Vol. 31, pp. 321–326, 2000.

[15] G.J. Schick, R.W. Wolverton, *An Analysis of Competing Software Reliability Models*, IEEE Transactions on Software Engineering, SE-4, No. 2, pp. 104–120, 1978.

[16] M.L. Shooman, *Probabilistic Models for Software Reliability Prediction*, Statistical Computer Performance Evaluation. Academic Press, New York, 1972.

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

[18] K. Tokuno, S. Yamada, *An imperfect debugging model with two types of hazard rates for software reliability measurement and assessment*, Mathematical and Computer Modelling, Vol. 31, pp. 343–352, 2000.

[19] M. Trachtenberg, *A general theory of software reliability modeling*, IEEE Transactions on Software Engineering, Vol. 39, No. 1, pp. 92–96, 1990.

[20] J.A. Whittaker, K. Rekab, M.G. Thomason, *A Markov chain model for predicting the reliability of multi-build software*, Information and Software Technology, Vol. 42, pp. 889–894, 2000.

[21] K. Worwa, *Estimation of the program testing strategy. Part 1 – The same errors can be encountered*, Cybernetics Research and Development, No. 3-4, pp. 155–173, 1995.

[22] K. Worwa, *Estimation of the program testing strategy. Part 2 – The same errors can not be encountered*, Cybernetics Research and Development, No. 3-4, pp. 175–188, 1995.

[23] K. Worwa, *Modelling and estimation of software reliability growth during the testing process*, Publishers of Warsaw Technical University, Warsaw (in Polish), 2000.

[24] S. Yamada, T. Fujiwara, *Testing-domain dependent software reliability growth models and their comparisons of goodness-of-fit*, International Journal of Reliability, No. 3, pp. 205–218, 2001.

[25] M.C. Yang, A. Chao, *Reliability-estimation & stopping-rules for software testing*, based on repeated appearances of bugs. IEEE Transactions on Reliability, Vol. 44, No. 2, pp. 315–321, 1995.

[26] X. Zhang, H. Pham, *Comparisons of nonhomogeneous Poisson process software reliability models and its applications*, International Journal of Systems Science, No. 9, 2000, pp. 1115–1123.

MODELOWANIE PROCESU TESTOWANIA OPROGRAMOWANIA

Streszczenie. Przedmiotem zawartych w artykule rozważań jest modelowanie procesu testowania programu, ze szczególnym uwzględnieniem modelowania wzrostu niezawodności programu w procesie jego testowania. W rozpatrywanym modelu testowany program charakteryzowany jest za pomocą tzw. macierzy charakterystycznej programu. Na bazie skonstruowanego modelu wyprowadzona została zależność na wartość oczekiwaną liczby błędów, wykrycie których spodziewane jest w wyniku realizacji procesu testowania, realizowanego w oparciu o przyjętą strategię testowania. Otrzymana zależność może być wykorzystana w praktyce, jeżeli macierz charakterystyczna programu jest znana. Dla przypadku, gdy macierz ta nie jest znana skonstruowane zostało w artykule obustronne oszacowanie tej wartości oczekiwanej.

Słowa kluczowe: testowanie oprogramowania, niezawodność oprogramowania, modele niezawodności oprogramowania.