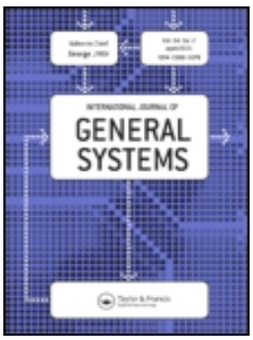

International Journal of General Systems

ISSN: 0308-1079 (Print) 1563-5104 (Online) Journal homepage: https://www.tandfonline.com/loi/ggen20

A mathematical modeling framework for software reliability testing

Kai-Yuan Cai , Zhao Dong , Ke Liu & Cheng-Gang Bai

To cite this article: Kai-Yuan Cai , Zhao Dong , Ke Liu & Cheng-Gang Bai (2007) A mathematical modeling framework for software reliability testing , International Journal of General Systems, 36:4, 399-463, DOI: 10.1080/03081070600957939

To link to this article: https://doi.org/10.1080/03081070600957939

International Journal of General Systems, Vol. 36, No. 4, August 2007, 399–463

Taylor & Francis
Taylor & Francis Group

A mathematical modeling framework for software reliability testing†

KAI-YUAN CAI‡*, ZHAO DONG¶, KE LIU¶ and CHENG-GANG BAI‡

‡Department of Automatic Control, Beijing University of Aeronautics and Astronautics, Beijing 100083, People's Republic of China
¶Institute of Applied Mathematics, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, Beijing 100080, People's Republic of China

(Received 21 December 2005; revised 19 July 2006; in final form 25 July 2006)

Software reliability testing refers to various software testing activities that are driven to achieve a quantitative reliability goal given *a priori* or lead to a quantitative reliability assessment for the software under test. In this paper we develop a modeling framework for the software reliability testing process, comprising a simplifying model and a generalized model. In both models the software testing action selection process and the defect removal mechanism are explicitly described. Both the discrete-time domain and the continuous-time domain are involved. The generalized model is more accurate or realistic than the simplifying model since the former avoids the assumption that defects are equally detectable and the assumption that defects are removed upon being detected. However simulation examples show that the simplifying model really captures some of essential features of the software testing process after a short initial testing stage. The modeling framework is practically realistic, mathematically rigorous, and quantitatively precise. It demonstrates that the relationship between software testing and delivered software reliability, which was poor understood, can well be formulated and quantified. Rigorous examinations show that several common assumptions adopted in software reliability modeling, including the independence assumption, the exponentiality assumption, and the NHPP assumption, are theoretically false in general. This paper sets a good starting point to further formalize and quantify the software testing process and its relation to delivered software reliability.

Keywords: Software testing; Software reliability; Markov usage model based testing; Software testing stability

1. Introduction

Software testing has long been one of major paradigms for achieving and validating software reliability (Pressman 2000). In a general sense, software testing is carried out throughout the whole life cycle of software development. It can be performed for reliability improvement by detecting and removing defects remaining in software document and code, or for reliability validation by freezing software document and code without removing remaining defects

*Corresponding author. Email: kycai@buaa.edu.cn

†Cai and Bai were supported by the National Natural Science Foundation of China (Grant Nos: 60233020, 60474006 and 60473067); Dong and Liu were partially supported by the National Natural Science Foundation of China.

International Journal of General Systems
ISSN 0308-1079 print/ISSN 1563-5104 online © 2007 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/03081070600957939

during testing. It can also be performed to assess the defect detection capability of a given test suite as demonstrated in mutation testing. However it was argued that software testing was the least understood part of software development process (Whittaker 2000). Software testing is mainly an ad hoc process and heavily depends on tester's knowledge and experience. Although there has been research work that addresses the effects of testing on software reliability (Malaiya and Karcich 1994, Chen *et al.* 1995, Frankl *et al.* 1998), the quantitative relationships between software testing processes and the delivered software reliability are not clear. This is very unsatisfactory since software testing is aimed for reliability improvement and/or reliability assessment. Software testing processes are far beyond being formalized, quantified or optimized. The major aim of this paper is to formalize the software testing process and quantify its effects on software reliability behavior, or simply, to formalize and quantify the software reliability testing process. By software reliability testing we mean software testing that is driven to achieve a quantitative reliability goal given *a priori* or leads to a quantitative reliability assessment for the software under test.

1.1 Related work

In order to formalize and quantify the software reliability testing process, we first note that there have been various approaches and techniques for software testing such as functional testing, control flow testing, data flow testing, and finite state machine testing (Beizer 1990, Zhu *et al.* 1997, Binder 2000). Most of them are mainly aimed to achieve certain test coverage or to satisfy certain test adequacy criteria. In these test coverage or adequacy criteria, software defects or reliability is not explicitly incorporated or quantified and thus the corresponding testing processes can hardly be treated as a software reliability testing process. The quantitative effects of these testing techniques on software reliability are not clear. On the other hand, the controlled Markov chain (CMC) approach to software testing (Cai 2002, Cai *et al.* 2004, 2005) can be treated as an approach for software reliability testing, where the feedback mechanisms in software testing are formalized, quantified and optimized for reliability improvement and/or reliability assessment from the perspective of test case selection during testing. However the quantitative effects of the CMC approach on software reliability have not been examined yet. Random testing is the most popular testing technique that is closely related to software reliability testing. For example, in the so-called software reliability engineered testing (Musa 1996), test cases are selected from the input domain of the software under test in accordance with the expected operational profile that is described by a probability distribution with the hope that the operational software reliability can accurately be estimated. Another form of random testing is the Markov usage model based testing (Whittaker and Poore 1993, Whittaker and Thomason 1994), by which test cases are executed in accordance with a Markov chain to reflect the interactive nature of the software under test. The testing data are then used to quantify the reliability improvement process and the delivered reliability. Obviously, conventional random testing or software reliability engineered testing is a special form of the Markov usage based testing.

Large amount of work has been reported on the Markov usage model based testing, which falls into several categories as follows.

(1) Usage modeling. Software usage can be modeled as a Markov chain (Whittaker and Poore 1993, Whittaker and Thomason 1994). It can also be modeled as Poisson process (Cai 1998, 2000), binary Markov process (Chen and Mills 1996), and non-Markov process (Doerner and Gutjahr 2000). Of course, the uniform probability distribution usually adopted in conventional random testing is a special case of Markov chain.
(2) State transition probability assignment. In order to carry out Markov usage model based testing, the state transition probabilities in the Markov usage model must be assigned. A modest amount of research works is devoted to this topic (Walton 1995, Semmel and Linton 1998, Gutjahr 2000, Walton and Poore 2000, Poore *et al.* 2000).
(3) Test case generation. Given a Markov usage model, an important problem is how to define and generate the required or optimal test cases. A simple way is to define every single state transition as a test case. An alternative way is to define a sequence of state transitions from the given initial state to the given final state. How to define and generate test cases has vast impacts on the effectiveness and efficiency of software testing and this is addressed in references (Sayre and Poore 2000a, Doerner and Gutjahr 2003, Beyer *et al.* 2003).
(4) Reliability assessment. An important advantage of Markov usage model based testing is that the resulting testing results can easily be adopted in the framework of Markov chain to quantitatively assess the delivered software reliability. Related works can be found in references (Gutjahr 1997, Whittaker *et al.* 2000, Sayre and Poore 2002).
(5) Stopping rules. Software testing must be stopped at right times. However it is difficult to determine the stopping times in an objective manner. In Markov usage model based testing, the discrepancy between the usage process and the resulting testing process is often used to define a stopping rule. References (Sahinoglu *et al.* 1999, Sayre and Poore 2000b) are devoted to the software testing stopping problem.

In addition, Avritzer and Weyuker discussed how to test software that was modeled as a Markov chain (Avritzer and Weyuker 1995).

From the perspective of software reliability testing, however, we can see that several drawbacks are associated with existing work on the Markov usage model based testing. First, software defect removal mechanisms are not explicitly considered. Software defects may be removed immediately upon being detected. They may remain un-removed until a given number of tests are executed or a given number of defects are revealed. They may also remain un-removed during testing. Different removal mechanisms have different impacts on software reliability behavior and this has extensively been examined in software reliability modeling (Xie 1991, Lyu 1996, Cai 1998). Second, two commonly adopted reliability measures, the times between successive observed failures and the total number of observed failures up to given instants of time, are seldom analyzed in a mathematically rigorous manner. An exception is the work of Thomason and Whittaker (1999), which shows that the two measures approach an exponential law and a Poisson law, respectively, as time proceeds for rare failure state in a Markov chain. However this work does not take account of the effects of test cases. Different test cases should differ in the capability of revealing failure and thus the transition probabilities from a normal software state to a failure state should vary with executed test cases. Third, the continuous-time domain is avoided. Test cases are selected and executed one by one and it is true that the effects of software testing on software reliability can be modeled in the discrete-time domain (Cai 2000). However executions of

different test cases take different CUP or calendar times and it should be more appropriate to combine the discrete-time domain with the continuous-time domain. The software reliability behavior should be examined in the discrete-time domain as well as in the continuous-time domain. The work of Semmel and Linton observed that different test cases took different continuous times during execution and showed how this information could be employed to determine the transition probabilities of the Markov usage model for software testing (Semmel and Linton 1998). It does not analyze the resulting behavior of software reliability in the continuous-time domain.

From the perspective of software reliability modeling, a major drawback of existing works on the Markov usage model based testing in particular, and on software reliability testing in general, is that they seldom examine in a mathematically rigorous manner if the common assumptions adopted in software reliability modeling are valid or not. These assumptions include:

(1) The independence assumption: the times between successive observed failures are independent;
(2) The exponentiality assumption: the times between successive observed failures are exponentially distributed; and
(3) The NHPP assumption: the cumulative number of observed failures follows a non-homogeneous Poisson process.

1.2 Desirable modeling framework

From the above analysis it is not surprising that existing works on software reliability testing do not well address the quantitative relationship between software testing and software reliability. It is not clear how software testing quantitatively affects software reliability behavior and how quantitative software reliability goals can be achieved through testing. In response to the undesirable current research status, this paper presents a mathematical modeling framework for software reliability testing. The central problem of concern is how to combine software reliability improvement with software testing process in a realistic, rigorous and quantitative manner. More specifically, the following requirements should be satisfied in a desirable modeling framework:

(1) Software reliability improvements are combined with software testing process and are explicitly characterized as software testing proceeds;
(2) Quantitative relationships between software reliability and software testing are described;
(3) The modeling framework is practically realistic, it well describes the engineering practice of software testing; and
(4) The modeling framework is mathematically rigorous so that in-depth theoretical results can be obtained.

Consequently, in the modeling framework presented in this paper, software defect removal mechanisms are explicitly formulated, the effects of software testing on software reliability behavior are rigorously analyzed, and both the discrete-time domain and continuous-time domain are adopted. Several mathematical propositions are obtained for software reliability, and the possibility of introducing a notion of stability for software testing is discussed.

1.3 Organization of the paper

The rest of this paper is organized as follows. Section 2 presents a simplifying mathematical model for the software reliability testing process. This model is closely related to the Markov usage model based testing and this is why Section 1.1 reviews related works on the Markov usage model based testing. However, it differs from existing models in that it explicitly models the defect removal mechanism and quantifies the relationship between software testing process and software reliability behavior. Both discrete-time domain and continuous-time domain are involved. Various properties of the simplifying model are examined in Sections 3–8. Section 3 presents a classification scheme for the cumulative number of observed failures and examine under what conditions it can form a Markov process or a higher-order Markov process. Although the cumulative number of observed failures does not form a Markov chain in general, Section 4 shows that the bi-variate process of the cumulative number of observed failures and the testing action taken in the course of software testing is a Markov process. This offers a convenient framework to investigate more properties of the software reliability testing process. Section 5 investigates several special cases in which the cumulative number of observed failures can be a time-homogeneous Markov process and/or the times between successive observed failures are exponentially distributed. Section 6 discusses the possibility of introducing a new notion of stability including inner stability and outer stability for software testing. Section 7 examines the expected behavior of the cumulative number of observed failures and shows that it follows or tends to follow an exponential law. Section 8 identifies various distinctions or specific time instants of concern in the software testing process and examines the behavior of times between successive distinctions. In Section 9 a generalized model is introduced and several simulation results are presented to demonstrate the behavior of software reliability testing. This new model is aimed to alleviate the unrealistic assumptions introduced in the simplifying model presented in Section 2 and is thus closer to the realistic software testing process. General discussion is presented in Section 10. Concluding remarks are contained in Section 11. Two mathematical propositions are included in the Appendix.

2. A simplifying model of software reliability testing

In software reliability testing test cases are selected one by one from the input domain or the given test suite of the software under test. Each test case may or may not reveal failures and detect software defects. The detected software defects, if any, are removed one by one and thus the reliability of the software under test demonstrates a growth trend. In this section we introduce a simplifying mathematical model to describe the process of test case selection and reliability growth.

2.1 Mathematical notation

Some of major mathematical notation that is consistently used throughout the rest of the paper is listed as follows. However this list is not complete. Section specific mathematical notation will be introduced in appropriate places in the rest of this paper.

C: the input domain or the given test suite of the software under test.

C_j: the jth class of C, $j = 1, 2, \ldots, m$; an class comprises a number of distinct test cases†, and it holds $C = \bigcup_{j=1}^{m} C_j$.

A_i: the ith testing action taken since the beginning of software testing, $i = 1, 2, \ldots$; $\forall i, A_i \in \{1, 2, \ldots, m\}$, and $A_i = j$ means that the ith testing action picks up a test case from C_j.

Z_i: indicator of failure revealed by A_i.

M_i: total number of failures revealed by $A_1, A_2, \ldots, A_i$.

$A(t)$: the testing action or the test case that is executed at time instant t, $t \in [0, \infty)$; $A(t) \in \{1, 2, \ldots, m\}$ and $A(t) = j$ means that a test case picked up from C_j is executed at time instant t.

$M(t)$: total number of failures revealed during the time interval $[0, t]$.

$H(t)$: total number of testing actions taken during the time interval $[0, t)$, excluding the first testing action that is taken at the beginning of testing.

N: total number of defects remaining in the software under test at the beginning of testing.

λ: testing intensity

p_{ij}: transition probability from state i to state j in a Markov chain.

Q: infinitesimal generator of a Markov process.

Remarks.

(1) In software testing the m classes $C_1, C_2, \ldots, C_m$ may or may not be disjoint.
(2) There is an one-to-one correspondence between testing actions and classes. $A_i = j$ or $A(t) = j$ means that a test case is picked up from C_j and is executed. Therefore the terms testing action (or action, simply) and class can interchangeably be used.
(3) $\{A_i, Z_i, M_i\}$ describes the software testing process in the discrete-time domain, whereas $\{A(t), M(t), H(t)\}$ describes the software testing process in the continuous-time domain.

2.2 Model assumptions

We have the following assumptions for the software testing process and the software defect removal mechanism.

(1) The input domain or the given test suite, C, of the software under test comprises m classes of test cases, $C_1, C_2, \ldots, C_m$, which may or may not be disjoint; that is, $C = \bigcup_{j=1}^{m} C_j$; $C_1, C_2, \ldots, C_m$ do not change in the course of software testing.
(2) The software under test contains N defects at the beginning of testing.
(3) Each test case picked up by an action or from a class may or may not reveal a failure; let

$$Z_i = \begin{cases} 1 & \text{if the } i\text{th action } A_i \text{ reveals a failure} \\ 0 & \text{otherwise} \end{cases}$$

(4) Upon a failure being revealed, the execution of the current test case terminates; one and only one failure-causing defect is removed immediately from the software under test, and no new defects are introduced.

†An alternative term adopted in the literature is "subdomain".

(5) A next test case is selected and executed after the current action is finished; the sequence $\{A_1, A_2, \ldots, A_i, A_{i+1}, \ldots\}$ forms a Markov chain with

$$\Pr\{A_{i+1} = l | A_i = k\} = p_{kl}.$$

(6) During the time interval $[0, t)$ a total of $H(t) + 1$ test cases are selected; the first one is taken at the beginning of software testing and $H(t)$ forms a Poisson process with parameter λ, or

$$\Pr\{H(t) = k\} = \frac{(\lambda t)^k}{k!} e^{\lambda t}; \quad k = 0, 1, 2, \ldots$$

where λ is referred to as the testing intensity, which represents the average number of tests or actions performed in an unity of time; each testing action including the first one takes an exponentially distributed length of time with parameter λ.

(7) The first i actions or test cases detect M_i defects and the testing process during the time interval $[0, t]$ detects $M(t)$ defects; that is,

$$M_i = \sum_{k=1}^{i} Z_k \quad \text{with } M_0 = 0$$

$$M(t) = \sum_{k=0}^{H(t)} Z_k \quad \text{with } M(0) = 0, \quad Z_0 = 0$$

(8) The probability of a test case revealing a failure is proportional to the number of defects remaining in the software under test,

$$\Pr\{Z_i = 1 | A_i = j, M_{i-1} = k\} = (N - k)\theta_j$$

$$\Pr\{Z_i = 0 | A_i = j, M_{i-1} = k\} = 1 - (N - k)\theta_j; \quad i = 1, 2, \ldots; \quad k = 0, 1, \ldots, N - 1$$

(9) $\{M_1, M_2, \ldots\}$ and $\{A_1, A_2, \ldots\}$ are conditionally independent of each other as follows,

$$\Pr\{M_1, A_2 | A_1\} = \Pr\{M_1 | A_1\} \Pr\{A_2 | A_1\}$$

and for $i > 1$,

$$\Pr\{M_i, A_{i+1} | M_{i-1}, \ldots, M_1, A_i, A_{i-1}, \ldots, A_1\}$$

$$= \Pr\{M_i | M_{i-1}, \ldots, M_1, A_i, A_{i-1}, \ldots, A_1\} \Pr\{A_{i+1} | M_{i-1}, \ldots, M_1, A_i, A_{i-1}, \ldots, A_1\}$$

$$\Pr\{M_i | M_{i-1}, \ldots, M_1, A_i, A_{i-1}, \ldots, A_1\} = \Pr\{M_i | M_{i-1}, A_i\}$$

$$\Pr\{A_{i+1} | M_{i-1}, \ldots, M_1, A_i, A_{i-1}, \ldots, A_1\} = \Pr\{A_{i+1} | A_i\}$$

(10) The process $\{M_i; i = 0, 1, 2, \ldots\}$ is independent of the Poisson process $\{H(t), t \geq 0\}$; more accurately, it holds

$$\Pr\{M_0 = 0, M_1 = k_1, \ldots, M_i = k_i | H(t) = i\} = \Pr\{M_0 = 0, M_1 = k_1, \ldots, M_i = k_i\}$$

(11) The process $\{A_i; i = 0, 1, 2, \ldots\}$ is independent of the Poisson process $\{H(t), t \geq 0\}$; more accurately, it holds

$$\Pr\{A_1 = j_1, A_2 = j_2, \ldots, A_i = j_i | H(t) = i\} = \Pr\{A_1 = j_1, A_2 = j_2, \ldots, A_i = j_i\}$$

(12) The first testing action is selected according to the probability distribution $\{p_1, p_2, \ldots, p_m\}$, that is, $\Pr\{A_1 = j\} = p_j$; $j = 1, 2, \ldots, m$.
(13) The software testing process terminates till all the N defects are removed.

Remarks.

(1) Each class comprises a number of test cases. When a test case is selected from a given class, we usually mean that it is selected from the given class at random. However no specific selection mechanism is further given in the above assumptions.
(2) According to the above assumptions, software testing proceeds as follows. At the beginning of software testing, $M_0 = 0$ and $M(0) = 0$. The first test case is selected from an class of the input domain or the given test suite of the software under test. Suppose it is selected from C_i, then $A_1 = i$. The test case is executed against the software under test. If a failure is revealed, then $Z_1 = 1$ and $M_1 = 1$, and one and only one failure-causing defect is removed; otherwise $Z_1 = 0$ and $M_1 = 0$. The execution of the first test case takes a total of exponentially distributed length of time with parameter λ (according to Assumption (6)). The second test case is selected from C_j with probability p_{ij}.
(3) $H(t) = 0$ implies that during the time interval $[0, t)$ the first action is executed, but has not been finished. Then according to Assumption (7), $M(s) \equiv Z_0 = 0$ for all $s < t$. $H(t) = 1$ implies that the first action has been finished and the second action is being executed. In this way, $M(t) \equiv Z_1$. In particular, if the first action is finished at the time instant $t = h_1$, then $M(s) \equiv Z_0 = 0$ for all $s < h_1$ and $M(h_1) = Z_1$.
(4) The software testing process can more precisely be described as follows. At the beginning of software testing the first action or test case is selected. This is finished instantaneously. Then the required test initialization is conducted. In general, the test initialization for an action may include setting initial state for the software under test, removing a failure-causing defect that was detected by the last action, and so on. The test initialization is finished instantaneously. This is followed by execution of the selected action or test case that takes an exponentially distributed length of time with parameter λ. The execution of the action then terminates. The termination is finished instantaneously. After termination, a test oracle or the tester checks or decides if a failure is revealed. The check is finished instantaneously. This completes the first action. Then the next action starts with a test case is selected, followed by the required test initialization and so on. We can see that an action actually comprises several subactions including *selection, initialization, execution, termination,* and *check*. All the subactions other than the execution are finished instantaneously. $A(t) = j$ means that a subaction corresponding to a test case of C_j is being conducted at time t.
(5) Mathematically, we can identify three distinct time instants $t-$, t and $t+$. Suppose that the jump points of the Poisson process $\{H(t), t \geq 0\}$ are $h_0, h_1, h_2, \ldots$. $h_0 = 0$, $h_1 = \inf\{t > 0 | H(t) = 1\}$, $\ldots$, $h_k = \inf\{t > h_{k-1} | H(t) = k\}$. Then we can have the following intuitive interpretation for h_1. The software testing process starts at time instant $t = 0+$ with the first action A_1 being selected. The first action terminates at time instant h_1-. Then the tester or a test oracle instantaneously checks at time instant h_1 if a failure is revealed. Consequently, $M(h_1) = M_1$ and we denote $Z(h_1) = Z_1$. Since the first action is yet to be completed, we have $A(h_1) = A(h_1-) = A_1$. At the time instant

h_1+ a new action is selected and thus $A(h_1+) = A_2$. Figure 1 depicts the process of software testing.

(6) A Poisson process describes the sum of several independent exponentially distributed random variables. It is chosen for $H(t)$ for the following reasons. First, although selections of actions at distinct instants of time follow a Markov chain, there is no evidence that the lengths of execution times for selected actions are correlated with each other. Actually, identical actions can be selected at distinct instants of time. It is natural to assume that the lengths of execution times for selected actions are independent. However this by no means that the times between successive software failures are independent. Second, the exponential distribution is determined by only one parameter and there is no evidence that more than one parameter is necessary for the length of execution time of a single action. The exponential distribution is a natural choice. Finally, Poisson process is a simple mathematical model which possesses many interesting mathematical properties.

(7) In general, we have

$$A(t) = \begin{cases} A_1 & \text{if } h_0 < t \le h_1 \\ A_2 & \text{if } h_1 < t \le h_2 \\ \cdots & \\ A_k & \text{if } h_{k-1} < t \le h_k \end{cases}$$

$$M(t) = \begin{cases} M_0 & \text{if } h_0 \le t < h_1 \\ M_1 & \text{if } h_1 \le t < h_2 \\ \cdots & \\ M_{k-1} & \text{if } h_{k-1} \le t < h_k \end{cases}$$

$$Z(t) = \begin{cases} Z_0 = 0 & \text{if } h_0 \le t < h_1 \\ Z_1 = M_1 - M_0 & \text{if } h_1 \le t < h_2 \\ \cdots & \\ Z_{k-1} = M_{k-1} - M_{k-2} & \text{if } h_{k-1} \le t < h_k \end{cases}$$

These equations establish fundamental relationships between the discrete-time domain and the continuous-time domain. Here $A(0)$ is undefined and we assume $A(0) = A(0+)$. However in general for $t > 0$, it holds $A(t) = A(t-)$. For $t \ge 0$, it holds $M(t) = M(t+)$.

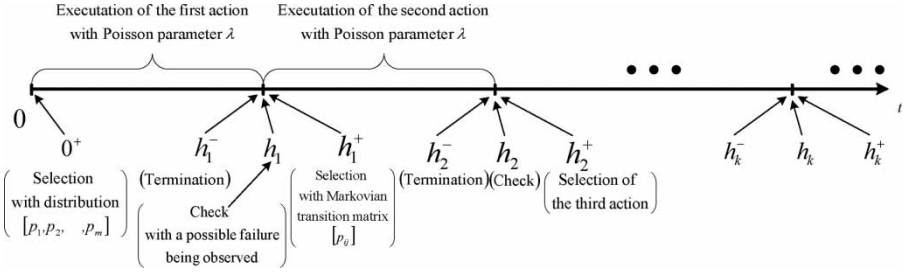

Figure 1. Software testing process.

(8) As a result of $A(t) = A(t-)$ for $t > 0$ and that there may be $A(t) \neq A(t+)$, we use $H(t) + 1$ to represent the total number of test cases selected during the time interval $[0, t)$ instead of during the time interval $[0, t]$. Similarly, as a result of $M(t) = M(t+)$ for $t \geq 0$, we use $M(t)$ to represent the total number of failures revealed during the time interval $[0, t]$ instead of during the time interval $[0, t)$. More specifically, we have $M(t) = M_{H(t)}$ and $A(t) = A_{H(t)+1}$.

(9) Assumption (4) implies that that there are N failures revealed in total. Each defect corresponds to one failure. Or equivalently, each test case or action detects at most one defect.

(10) A test case should widely be interpreted. For a transformational or scientific program that computes the function $f(x)$, a test case may be defined as an input value assigned to the variable x. For a GUI program that can be modeled as a finite state machine, it may be defined as an initial state plus a sequence of state transitions. More definitions can be adopted.

(11) In Assumption (9), an abbreviated notation is employed. In $\Pr\{M_i, A_{i+1} | M_{i-1}, \ldots, M_1, A_i, A_{i-1}, \ldots, A_1\}$, M_k can take any value from $\{0, 1, \ldots, k\}$, $k = 1, 2, \ldots, i$, and $A_1, A_2, \ldots, A_{i+1}$ can take any value from $\{1, 2, \ldots, m\}$.

(12) Assumption (9) mathematically characterizes the conditional independence between $\{M_1, M_2, \ldots\}$ and $\{A_1, A_2, \ldots\}$. It actually implies that given the current action A_1 (say) being executed, the results of two consecutive subactions, check (for the current action) and selection (for the next action), are independent.

(13) Simply speaking, the software testing process is determined by $\{\theta_1, \theta_2, \ldots, \theta_m\}$, $\lfloor p_{ij} \rfloor_{m \times m}$ and $\{\Pr\{A_1 = j\}, j = 1, 2, \ldots, m\}$. We can say that the three sets of parameters fully define a testing strategy mathematically.

2.3 Threats to the validity of the model assumptions

There are several threats that may invalidate the above model assumptions.

(1) Assumption (4) states that a failure-causing defect is removed immediately upon the failure being revealed. This may be true in unit testing. However this is not true in general. In integration or system testing, defect removals are carried out only after a given number of tests are executed or a given number of defects are detected. We will revisit this in Section 9.

(2) Assumption (4) also states that one and only one failure-causing defect is removed for each revealed failure. This may or may not be true. It is possible for a single test case to detect more than one defect and thus several defects may be removed as a result of a revealed failure.

(3) Assumption (8) states that all the remaining defects are equally detectable, although different actions may have different defect detection probabilities. This assumption is commonly adopted in software reliability modeling for the sake of mathematical tractability (Xie 1991, Lyu 1996, Cai 1998), however it is generally false. Different defects have different probabilities of being detected. A realistic assumption is that defects can be divided into several clusters and defects in a single cluster are equally detectable. Defects in different clusters are not equally detectable. This will be taken into account in the generalized model presented in Section 9.

(4) Assumption (9) implicitly states that the initial number of defects remaining in the software under test is known. This is not true. In reality this number is not known and must be estimated and updated on-line.

Despite the various deviations from the software reliability testing practice, the proposed mathematical model is still valuable. The value of it lies in the fact that it puts the software testing process and the software reliability behavior together and formulates the quantitative relationship between them in a mathematically rigorous manner. In order to keep the required mathematical treatments manageable, we have to make trade-off between engineering reality and mathematical tractability and take simplifying assumptions. A generalized model presented in Section 9 is more mathematically involved and closer to the realistic software testing process. However the simulation results presented in Section 9 indicate that the simplifying model still makes sense. Of course, this by no means implies that the proposed mathematical models cannot be improved. This paper is only a starting point to develop more realistic modeling frameworks and to do decision-making for software testing and reliability.

2.4 Comparison with related models

Related mathematical models have been reviewed in Section 1. The mathematical model proposed in the above is distinctly different from them. This first class of related models is the Markov usage models developed for software testing (Whittaker and Poore 1993, Whittaker and Thomason 1994). These models guide the selections of various actions during software testing. However they do not consider how software testing quantitatively affects software reliability. Neither is the continuous-time domain involved.

The second class of related models is the software reliability (growth) models (Xie 1991, Lyu 1996, Cai 1998). An essential or implicit assumption in most of these models is that the test profile coincides with the operational profile. The effects of software test case generation and software testing are not explicitly considered.

The third class of related models is the CMC for software testing (Cai 2002, Cai *et al.* 2004, 2005). These models treat software testing as a control problem and show how to select test cases to achieve a quantitative testing (reliability) goal in an optimal manner. However these models do not analyze the behavior of the cumulative number of observed failures and the times between successive observed failures.

2.5 Problems of concern

Two problems are of concern and will be investigated in depth in the rest of this paper.

(1) How do $\{M_i, i = 1, 2, \dots\}$ and $\{M(t); t \geq 0\}$ behave? Do they form a Markov chain, a Markov process or a non-homogeneous Poisson process?
(2) How does $\{\tau_i, i = 1, 2, \dots\}$ behave? Are they independent and exponentially distributed? Here $\{\tau_i, i = 1, 2, \dots\}$ represents the times between successive observed failures and the like.

The first problem is related to the NHPP assumption mentioned in Section 1.1. A major class of existing software reliability models adopts the NHPP assumption. The second problem is related to the exponentiality assumption mentioned in Section 1.2. This assumption implies that the times between successive observed failures are exponentially distributed and was adopted in another major class of existing software reliability models. Since most existing software reliability models fail to deliver their promises for software reliability forecasting and software process improvement, it is important to examine if and why these common assumptions are valid or invalid.

3. A classification scheme for $\{M_i, i = 1, 2, \dots\}$

In this section, we first show that $\{M_i, i = 1, 2, \dots\}$ is not a Markov chain in general. We then provide a classification scheme for $\{M_i, i = 1, 2, \dots\}$ to determine if it can be a higher-order Markov chain.

In order to check if $\{M_i, i = 1, 2, \dots\}$ is a Markov chain, we need to calculate $\Pr\{M_{i+1}|M_i, M_{i-1}, \dots, M_1, M_0\}$. Note that

$$\Pr\{M_{i+1}|M_i, M_{i-1}, \dots, M_1, M_0\} = \frac{\Pr\{M_{i+1}, M_i, \dots, M_1, M_0\}}{\Pr\{M_i, M_{i-1}, \dots, M_1, M_0\}}$$

$$\Pr\{M_{i+1}, M_i, \dots, M_1, M_0\} = \sum_{A_1, \dots, A_{i+1}} \Pr\{M_{i+1}, M_i, \dots, M_1, M_0, A_{i+1}, A_i, \dots, A_1\}$$

Further note that $\Pr\{M_0 = 0\} = 1$. We can calculate $\Pr\{M_{i+1}, M_i, M_{i-1}, \dots, M_1, M_0, A_{i+1}, A_i, \dots, A_1\}$ in a recursive manner as follows.

Step 1 For $\Pr\{M_1, M_0, A_1\}$ we have

$$\Pr\{M_1, M_0, A_1\} = \Pr\{M_1|M_0, A_1\}\Pr\{A_1|M_0\} = \big(Z_1 N\theta_{A_1} + (1 - Z_1)\big(1 - N\theta_{A_1}\big)\big)p_{A_1}$$

where

$$Z_1 = M_1 - M_0$$

Therefore

$$\Pr\{M_1|M_0\} = \Pr\{M_1, M_0\} = \sum_{A_1}\Pr\{M_1, M_0, A_1\}$$

$$= \sum_{A_1}\big(Z_1 N\theta_{A_1} + (1 - Z_1)\big(1 - N\theta_{A_1}\big)\big)p_{A_1}$$

Step 2 In general, $\Pr\{M_{i+1}, M_i, M_{i-1}, \dots, M_1, M_0, A_{i+1}, A_i, \dots, A_1\}$ can be expressed in terms of $\Pr\{M_i, M_{i-1}, \dots, M_1, M_0, A_i, \dots, A_1\}$. In fact from Assumptions (8) and (9) we have

$$\begin{aligned}
&\Pr\{M_{i+1}, M_i, M_{i-1}, \dots, M_1, M_0, A_{i+1}, A_i, \dots, A_1\}\\
&= \Pr\{M_{i+1}|M_i, M_{i-1}, \dots, M_1, M_0, A_{i+1}, A_i, \dots, A_1\}\\
&\quad \times \Pr\{M_i, M_{i-1}, \dots, M_1, M_0, A_{i+1}, A_i, \dots, A_1\}\\
&= \Pr\{M_{i+1}|M_i, A_{i+1}\}\Pr\{M_i, M_{i-1}, \dots, M_1, M_0, A_{i+1}, A_i, \dots, A_1\}\\
&= \Pr\{M_{i+1}|M_i, A_{i+1}\}\Pr\{M_i, A_{i+1}|M_{i-1}, \dots, M_1, M_0, A_i, \cdots, A_1\}\\
&\quad \times \Pr\{M_{i-1}, \dots, M_1, M_0, A_i, \dots, A_1\}\\
&= \Pr\{M_{i+1}|M_i, A_{i+1}\}\Pr\{A_{i+1}|M_{i-1}, \dots, M_1, M_0, A_i, \dots, A_1\}\\
&\quad \times \Pr\{M_i|M_{i-1}, \dots, M_1, M_0, A_i, \dots, A_1\}\Pr\{M_{i-1}, \dots, M_1, M_0, A_i, \dots, A_1\}\\
&= \Pr\{M_{i+1}|M_i, A_{i+1}\}\Pr\{A_{i+1}|A_i\}\Pr\{M_i, \dots, M_1, M_0, A_i, \dots, A_1\}\\
&= \big(Z_{i+1}(N - M_i)\theta_{A_{i+1}} + (1 - Z_{i+1})\big(1 - (N - M_i)\theta_{A_{i+1}}\big)\big)\\
&\quad \times p_{A_i A_{i+1}}\Pr\{M_i, \dots, M_1, M_0, A_i, \dots, A_1\}
\end{aligned}$$

where

$$Z_{i+1} = M_{i+1} - M_i$$

PROPOSITION 3.1. $\{\langle M_i, A_i\rangle; i = 1, 2, \dots\}$ is a Markov chain.

Proof. This is obvious from Step 2 presented above. □

PROPOSITION 3.2. $\{\langle M_i, A_i\rangle; i = 0, 1, \dots\}$ is a Markov chain.

Proof. From Assumption (9), for $i > 1$,

$$\Pr\{M_i, A_{i+1} | M_{i-1}, \dots, M_1, A_i, A_{i-1}, \dots, A_1\} = \Pr\{M_i | M_{i-1}, A_i\}\Pr\{A_{i+1} | A_i\}$$

In this way we have

$$\sum_{M_{i-2},\dots,M_1;A_{i-1},\dots,A_1} \Pr\{M_i,A_{i+1};M_{i-1},M_{i-2},\dots,M_1,A_i,A_{i-1},A_{i-2},\dots,A_1\}$$

$$= \sum_{M_{i-2},\dots,M_1;A_{i-1},\dots,A_1} \Pr\{M_i|M_{i-1},A_i\}\Pr\{A_{i+1}|A_i\}\Pr\{M_{i-1},M_{i-2},\dots,M_1,A_i,A_{i-1},A_{i-2},\dots,A_1\}$$

That is,

$$\Pr\{M_i, A_{i+1}; M_{i-1}, A_i\} = \Pr\{M_i | M_{i-1}, A_i\}\Pr\{A_{i+1} | A_i\}\Pr\{M_{i-1}, A_i\}$$

Therefore

$$\Pr\{M_i, A_{i+1} | M_{i-1}, A_i\} = \Pr\{M_i | M_{i-1}, A_i\}\Pr\{A_{i+1} | A_i\}$$

So

$$\Pr\{M_i, A_{i+1} | M_{i-1}, \dots, M_1, A_i, A_{i-1}, \dots, A_1\} = \Pr\{M_i, A_{i+1} | M_{i-1}, A_i\}$$

□

PROPOSITION 3.3. For the Markov chain $\{\langle M_i, A_{i+1}\rangle; i = 0, 1, \dots\}$, it holds

$$\Pr\{M_k = n, A_{k+1} = j | M_{k-1} = n, A_k = i\} = [1 - (N - n)\theta_i]p_{ij}$$
$$\Pr\{M_k = n + 1, A_{k+1} = j | M_{k-1} = n, A_k = i\} = (N - n)\theta_i p_{ij}$$

Proof. See Appendix. □

Although both $\{\langle M_i, A_i\rangle; i = 1, 2, \dots\}$ and $\{\langle M_i, A_{i+1}\rangle; i = 0, 1, \dots\}$ are a Markov chain, $\{M_i, i = 1, 2, \dots\}$ is not a Markov chain in general. We can show this by directly calculating $\Pr\{M_{i+1} | M_i, M_{i-1}, \dots, M_1, M_0\}$ for different assignments of $\{M_{i+1}, M_i, M_{i-1}, \dots, M_1, M_0\}$ with an example.

Example 3.1. Let

$$N = 30, \quad m = 2$$
$$\theta_1 = 0.007100, \quad \theta_2 = 0.020233$$
$$\Pr\{A_1 = 1\} = 0.421215$$
$$\Pr\{A_1 = 2\} = 0.578785$$
$$[p_{ij}] = \begin{bmatrix} 0.624862 & 0.375138 \\ 0.188154 & 0.811846 \end{bmatrix}$$

Then we have

$$\Pr\left\{ M_{10}=6 \middle| \begin{matrix} M_9=5, M_8=5, M_7=5, M_6=5, M_5=5, \\ M_4=4, M_3=3, M_2=2, M_1=1, M_0=0 \end{matrix} \right\} = 0.366920$$

$$\Pr\left\{ M_{10}=6 \middle| \begin{matrix} M_9=5, M_8=4, M_7=3, M_6=2, M_5=1, \\ M_4=0, M_3=0, M_2=0, M_1=0, M_0=0 \end{matrix} \right\} = 0.430447$$

This implies that $\{M_i, i=1,2,\dots\}$. Consequently, $\{M(t); t\geq 0\}$ is not a Markov process in general either. □

A natural question is if $\{M_i, i=1,2,\dots\}$ is a higher-order Markov chain or under what conditions it can be a higher-order Markov chain. We have the following Proposition.

PROPOSITION 3.4. Suppose it holds

$$\Pr\{A_i|M_i, M_{i-1}, M_{i-2}, \dots, M_1\} = \Pr\{A_i|M_i, M_{i-1}, \dots, M_k\}$$

Then

$$\Pr\{M_{i+1}|M_i, M_{i-1}, \dots, M_1\} = \Pr\{M_{i+1}|M_i, M_{i-1}, \dots, M_k\}$$

Proof. From Proposition 3.1, we have

$$\begin{aligned}
&\Pr\{M_{i+1}|M_i,M_{i-1},\dots,M_1\} \\
&=\frac{\Pr\{M_{i+1},M_i,M_{i-1},M_{i-2},\dots,M_1\}}{\Pr\{M_i,M_{i-1},M_{i-2},\dots,M_1\}} \\
&=\frac{\sum_{A_{i+1},A_i,A_{i-1},A_{i-2},\dots,A_1} \Pr\{M_{i+1},A_{i+1},M_i,A_i,M_{i-1},A_{i-1},\dots,M_1,A_1\}}{\Pr\{M_i,M_{i-1},M_{i-2},\dots,M_1\}} \\
&=\frac{\sum_{A_{i+1},A_i,A_{i-1},A_{i-2},\dots,A_1} \Pr\{M_{i+1},A_{i+1}|M_i,A_i,M_{i-1},A_{i-1},\dots,M_1,A_1\}\Pr\{M_i,A_i,M_{i-1},A_{i-1},\dots,M_1,A_1\}}{\Pr\{M_i,M_{i-1},M_{i-2},\dots,M_1\}} \\
&=\frac{\sum_{A_{i+1},A_i,A_{i-1},A_{i-2},\dots,A_1} \Pr\{M_{i+1},A_{i+1}|M_i,A_i\}\Pr\{M_i,A_i,M_{i-1},A_{i-1},\dots,M_1,A_1\}}{\Pr\{M_i,M_{i-1},M_{i-2},\dots,M_1\}} \\
&=\frac{\sum_{A_{i+1},A_i}\sum_{A_{i-1},A_{i-2},\dots,A_1} \Pr\{M_{i+1},A_{i+1}|M_i,A_i\}\Pr\{M_i,A_i,M_{i-1},A_{i-1},\dots,M_1,A_1\}}{\Pr\{M_i,M_{i-1},M_{i-2},\dots,M_1\}} \\
&=\frac{\sum_{A_{i+1},A_i} \Pr\{M_{i+1},A_{i+1}|M_i,A_i\}\sum_{A_{i-1},A_{i-2},\dots,A_1}\Pr\{M_i,A_i,M_{i-1},A_{i-1},\dots,M_1,A_1\}}{\Pr\{M_i,M_{i-1},M_{i-2},\dots,M_1\}} \\
&=\frac{\sum_{A_{i+1},A_i} \Pr\{M_{i+1},A_{i+1}|M_i,A_i\}\Pr\{M_i,A_i,M_{i-1},M_{i-2},\dots,M_1\}}{\Pr\{M_i,M_{i-1},M_{i-2},\dots,M_1\}} \\
&=\frac{\sum_{A_i}\Pr\{M_{i+1}|M_i,A_i\}\Pr\{M_i,A_i,M_{i-1},M_{i-2},\dots,M_1\}}{\Pr\{M_i,M_{i-1},M_{i-2},\dots,M_1\}} \\
&=\sum_{A_i}\Pr\{M_{i+1}|M_i,A_i\}\frac{\Pr\{M_i,A_i,M_{i-1},M_{i-2},\dots,M_1\}}{\Pr\{M_i,M_{i-1},M_{i-2},\dots,M_1\}} \\
&=\sum_{A_i}\Pr\{M_{i+1}|M_i,A_i\}\Pr\{A_i|M_i,M_{i-1},M_{i-2},\dots,M_1\}
\end{aligned}$$

□

The above Proposition actually defines a classification scheme for $\{M_i, i = 1, 2, \dots\}$. If it holds

$$\Pr\{A_i|M_i, M_{i-1}, M_{i-2}, \dots, M_1\} = \Pr\{A_i|M_i\}$$

that is, the probability of A_i can be inferred from M_i and is conditionally independent of $\{M_{i-1}, \dots, M_1\}$, then

$$\Pr\{M_{i+1}|M_i, M_{i-1}, \dots, M_1\} = \Pr\{M_{i+1}|M_i\}$$

or $\{M_i, i = 1, 2, \dots\}$ is a Markov chain. Note that the effect of A_i is represented by Z_i and thus is included in M_i. However M_i also accumulates the effects of $A_{i-1}, \dots, A_1$, and thus is not sufficient for A_i to be inferred in general. A special case for $\{M_i, i = 1, 2, \dots\}$ to be a Markov chain is that it holds

$$\Pr\{A_i|M_i, M_{i-1}, M_{i-2}, \dots, M_1\} = \Pr\{A_i|M_i\} = \Pr\{A_i\}$$

that is, the probability of an action being taken is independent of the testing history. This scenario corresponds to the random testing strategy that test cases are selected from the input domain or test suite of the software under test in accordance with a given probability distribution. In other words, conventional random testing generates a Markov chain $\{M_i, i = 1, 2, \dots\}$. The necessary condition for $\{M_i, i = 1, 2, \dots\}$ to be a Markov chain is an open problem.

If it holds

$$\Pr\{A_i|M_i, M_{i-1}, M_{i-2}, \dots, M_1\} = \Pr\{A_i|M_i, M_{i-1}\}$$

then

$$\Pr\{M_{i+1}|M_i, M_{i-1}, \dots, M_1\} = \Pr\{M_{i+1}|M_i, M_{i-1}\}$$

that is, $\{M_i, i = 1, 2, \dots\}$ is a two-order Markov chain.

4. Markovian properties of $\{\langle \boldsymbol{M_i}, \boldsymbol{A_i}\rangle; \boldsymbol{i = 1, 2, \dots}\}$ and $\{\langle \boldsymbol{M(t)}, \boldsymbol{A(t)}\rangle; \boldsymbol{t \geq 0}\}$

Since $\{M_i, i = 1, 2, \dots\}$ is not a Markov chain in general, it becomes difficult to investigate the behavior of it directly. Fortunately, Proposition 3.1 asserts that $\{\langle M_i, A_i\rangle; i = 1, 2, \dots\}$ is a Markov chain. More specifically, from Assumptions (8) and (9), the transition probability of $\langle M_i, A_i\rangle$ depending on $\langle M_{i-1}, A_{i-1}\rangle$ is

$$\begin{aligned}
&\Pr\{M_i = k, A_i = j|M_{i-1} = k, A_{i-1} = l\} \\
&\quad = \Pr\{M_i = k|A_i = j, M_{i-1} = k, A_{i-1} = l\}\Pr\{A_i = j|M_{i-1} = k, A_{i-1} = l\} \\
&\quad = \Pr\{Z_i = 0|A_i = j\}\Pr\{A_i = j|A_{i-1} = l\} = (1 - (N - k)\theta_j)p_{lj} \\
&\Pr\{M_i = k + 1, A_i = j|M_{i-1} = k, A_{i-1} = l\} \\
&\quad = \Pr\{M_i = k + 1|A_i = j, M_{i-1} = k, A_{i-1} = l\}\Pr\{A_i = j|M_{i-1} = k, A_{i-1} = l\} \\
&\quad = \Pr\{Z_i = 1|A_i = j\}\Pr\{A_i = j|A_{i-1} = l\} = (N - k)\theta_j p_{lj} \\
&\qquad \Pr\{M_i \notin \{k, k + 1\}, A_i = j|M_{i-1} = k, A_{i-1} = l\} = 0
\end{aligned}$$

These state transition probabilities completely specify the behavior of $\{\langle M_i, A_i\rangle; i = 1, 2, \dots\}$. The Appendix presents the corresponding transition probability matrices of $\{\langle M_i, A_i\rangle; i = 1, 2, \dots\}$ and $\{\langle M_i, A_{i+1}\rangle; i = 0, 1, \dots\}$.

PROPOSITION 4.1. $\{\langle M(t), A(t)\rangle; t \geq 0\}$ is a Markov process.

Proof. From Proposition 3.2 we know that $\{\langle M_i, A_{i+1}\rangle; i = 0, 1, \dots\}$ is a Markov chain. From Assumptions (10) and (11) $\{\langle M_i, A_{i+1}\rangle; i = 0, 1, \dots\}$ is independent of $\{H(t), t \geq 0\}$ that is a Poisson process $\{H(t), t \geq 0\}$ with parameter λ, and $\langle M(t), A(t)\rangle = \langle M_{H(t)}, A_{H(t)+1}\rangle$ (refer to Remark (7) of Section 2.2), we conclude that $\langle M(t), A(t); t \geq 0\rangle$ is a Markov process (Feller 1971). □

PROPOSITION 4.2. Suppose the jump points of the Poisson process $\{H(t), t \geq 0\}$ are $h_0, h_1, h_2, \dots$. $h_0 = 0$, $h_1 = \inf\{t > 0 | H(t) = 1\}$, $\dots$, $h_k = \inf\{t > h_{k-1} | H(t) = k\}$. Let f be a measurable real-valued function defined on $\{0, 1, \dots, N\} \times \{1, 2, \dots, m\}$. Then it holds

$$E_{\langle M(0), A(0+)\rangle} f(M(t), A(t)) = \sum_{k=0}^{\infty} E_{\langle M(0), A(0+)\rangle} f(M_k, A_{k+1}) \mathrm{e}^{-\lambda t} \frac{(\lambda t)^k}{k!}$$

where $E_{\langle M(0), A(0+)\rangle}$ denotes the mathematical expectation with respect to $P_{\langle M(0), A(0+)\rangle}$, which denotes the probability measure of the Markov process $\{\langle M(t), A(t)\rangle; t \geq 0\}$ with the initial state $\langle M(0), A(0+)\rangle$.

Proof. See Appendix. □

For $\{\langle M(t), A(t)\rangle; t \geq 0\}$, suppose we represent the state space as

$$\begin{bmatrix} \langle 0, \text{Action } 1\rangle & \langle 0, \text{Action } 2\rangle & \dots & \langle 0, \text{Action } m\rangle \\ \langle 1, \text{Action } 1\rangle & \langle 1, \text{Action } 2\rangle & \dots & \langle 1, \text{Action } m\rangle \\ \vdots & & & \\ \langle N, \text{Action } 1\rangle & \langle N, \text{Action } 2\rangle & \dots & \langle N, \text{Action } m\rangle \end{bmatrix}$$

Then we can see that state transitions can only occur within the same row or from a row to its next (adjacent) row. No other state transitions are possible. Now let us represent the state space as a single row

$$[\langle 0, \text{Action } 1\rangle \cdots \langle 0, \text{Action } m\rangle \langle 1, \text{Action } 1\rangle \cdots \langle 1, \text{Action } m\rangle \langle N, \text{Action } 1\rangle \cdots \langle N, \text{Action } m\rangle]$$

Note that in each state the software testing process stay for an exponentially distributed length of time with parameter λ. It then transforms to one of the $(N + 1)m$ distinct states with proper probabilities or rates. To write down the Kolmogorov-Chapman equation of $\{\langle M(t), A(t)\rangle; t \geq 0\}$, we let $P_{\langle c,l\rangle\langle b,j\rangle}(s, t)$ denote the probability that the testing process is in the state $\langle b, j\rangle$ at time instant t, given that the testing process is in state $\langle c, l\rangle$ at time instant s, $s < t$; that is,

$$P_{\langle c,l\rangle\langle b,j\rangle}(s, t) = \Pr\{M(t) = b, A(t) = j | M(s) = c, A(s) = l\}; 0 \leq s < t$$

For $c = 0$, and $b = 1, 2, \ldots, N-1$; $\quad j = 1, 2, \ldots, m$, we have

$$P_{\langle 0,l\rangle\langle b,j\rangle}(s,t+\Delta t) = (1-\lambda\Delta t)P_{\langle 0,l\rangle\langle b,j\rangle}(s,t) + \lambda\Delta t\sum_{k=1}^{m} P_{\langle 0,l\rangle\langle b,k\rangle}(s,t)[1-(N-b)\theta_k]p_{kj}$$
$$+ \lambda\Delta t\sum_{k=1}^{m} P_{\langle 0,l\rangle\langle b-1,k\rangle}(s,t)(N-b+1)\theta_k p_{kj} + o(\Delta t)$$

where $o(\Delta t)$ denotes higher-order infinitesimal of Δt. In this way

$$\frac{\mathrm{d}}{\mathrm{d}t}P_{\langle 0,l\rangle\langle b,j\rangle}(s,t) = -\lambda P_{\langle 0,l\rangle\langle b,j\rangle}(s,t) + \lambda\sum_{k=1}^{m} P_{\langle 0,l\rangle\langle b,k\rangle}(s,t)[1-(N-b)\theta_k]p_{kj}$$
$$+ \lambda\sum_{k=1}^{m} P_{\langle 0,l\rangle\langle b-1,k\rangle}(s,t)(N-b+1)\theta_k p_{kj}$$

Similarly, for $b = 0$ and $j = 1, 2, \ldots, m$, we have

$$\frac{\mathrm{d}}{\mathrm{d}t}P_{\langle 0,l\rangle\langle 0,j\rangle}(s,t) = -\lambda P_{\langle 0,l\rangle\langle 0,j\rangle}(s,t) + \lambda\sum_{k=1}^{m} P_{\langle 0,l\rangle\langle b,k\rangle}(s,t)[1-N\theta_k]p_{kj}$$

For $b = N$ and $j = 1, 2, \ldots, m$, we have

$$\frac{\mathrm{d}}{\mathrm{d}t}P_{\langle 0,l\rangle\langle N,j\rangle}(s,t) = \lambda\sum_{k=1}^{m} P_{\langle 0,l\rangle\langle N-1,k\rangle}(s,t)\theta_k p_{kj}$$

In summary, we obtain

$$\frac{\mathrm{d}P}{\mathrm{d}t} = P\cdot Q$$

where P denotes the *row vector*

$$P = \begin{bmatrix} P_{\langle 0,l\rangle\langle 0,1\rangle}(s,t), P_{\langle 0,l\rangle\langle 0,2\rangle}(s,t), \ldots, P_{\langle 0,l\rangle\langle 0,m\rangle}(s,t), \\ P_{\langle 0,l\rangle\langle 1,1\rangle}(s,t), P_{\langle 0,l\rangle\langle 1,2\rangle}(s,t), \ldots, P_{\langle 0,l\rangle\langle 1,m\rangle}(s,t), \\ \ldots, \\ P_{\langle 0,l\rangle\langle N,1\rangle}(s,t), P_{\langle 0,l\rangle\langle N,2\rangle}(s,t), \ldots, P_{\langle 0,l\rangle\langle N,m\rangle}(s,t), \end{bmatrix}$$

and Q denotes $Q_{(N+1)m\times(N+1)m}$, which is the infinitesimal generator of the Markov process $\{\langle M(t), A(t)\rangle; t \geq 0\}$,

$$Q_{(N+1)m\times(N+1)m}$$
$$= \begin{bmatrix} Q^1_{m\times m}(0) & Q^2_{m\times m}(0) & & & & \\ & Q^1_{m\times m}(1) & Q^2_{m\times m}(1) & & & \\ & & Q^1_{m\times m}(2) & Q^2_{m\times m}(2) & & \\ & & & \ddots & & \\ & & & & Q^1_{m\times m}(N-1) & Q^2_{m\times m}(N-1) \\ & & & & 0 & 0 \end{bmatrix}$$

where

$$Q^1_{mm}(k) = \begin{bmatrix} \lambda[1-(N-k)\theta_1]p_{11}-\lambda & \lambda[1-(N-k)\theta_1]p_{12} & \dots & \lambda[1-(N-k)\theta_1]p_{1m} \\ \lambda[1-(N-k)\theta_2]p_{21} & \lambda[1-(N-k)\theta_2]p_{22}-\lambda & \dots & \lambda[1-(N-k)\theta_2]p_{2m} \\ \vdots & \vdots & & \vdots \\ \lambda[1-(N-k)\theta_m]p_{m1} & \lambda[1-(N-k)\theta_m]p_{m2} & & \lambda[1-(N-k)\theta_m]p_{mm}-\lambda \end{bmatrix}$$

$$Q^2_{mm}(k) = \begin{bmatrix} \lambda(N-k)\theta_1 p_{11} & \lambda(N-k)\theta_1 p_{12} & \dots & \lambda(N-k)\theta_1 p_{1m} \\ \lambda(N-k)\theta_2 p_{21} & \lambda(N-k)\theta_2 p_{22} & \dots & \lambda(N-k)\theta_2 p_{2m} \\ \vdots & \vdots & & \vdots \\ \lambda(N-k)\theta_m p_{m1} & \lambda(N-k)\theta_m p_{m2} & & \lambda(N-k)\theta_m p_{mm} \end{bmatrix}$$

The infinitesimal generator fully characterizes the behavior of $\{\langle M(t), A(t)\rangle; t \geq 0\}$.

5. Special cases

Proposition 3.3 identifies a sufficient condition for $\{M_i, i = 1, 2, \dots\}$ to be a Markov chain or a higher-order Markov chain. However this sufficient condition is not easy to verify. Further, it does not tell under what condition the times between successive observed failures are exponentially distributed, although we assume that the times between successive terminations of taken actions are exponentially distributed with parameter λ. Note that $\{\langle M_i, A_i\rangle; i = 1, 2, \dots\}$ is a Markov chain and $\{\langle M(t), A(t)\rangle; t \geq 0\}$ is a Markov process. From this in this section we consider several special cases in which $\{M_i, i = 1, 2, \dots\}$ or $\{M(t); t \geq 0\}$ is a Markov chain and/or the times between successive observed failures are exponentially distributed.

5.1 Case 1: $p_{ij} \equiv p_j$; $i,j = 1, 2, \dots, m$

Proposition 5.1. Suppose it holds

$$p_{ij} \equiv p_j; \quad i,j = 1, 2, \dots, m$$

Then $\{M_i; i = 0, 1, 2, \dots\}$ is a time-homogeneous Markov chain.

Proof. See Appendix. □

Remarks.

(1) In Section 3 we have concluded that $\{M_i; i = 0, 1, 2, \dots\}$ under conventional random testing is a Markov chain. Proposition 5.1 further asserts that the Markov chain is time-homogeneous.

(2) Mathematically, a time-homogeneous process implies that for any initial state it starts, the subsequent behavior is a function of the time discrepancy between the initial state and the current state and is independent of the time instant the process starts. However the process $\{M_i; i = 0, 1, 2, \dots\}$ of our concern always starts with $M_0 = 0$. $l \geq k_2$ is required as a result of the assumptions presented in Section 2.2. Each action can reveal at most one failure and thus $M_l \leq l$.

COROLLARY 5.1. Suppose it holds

$$p_{ij} \equiv p_j; \quad i,j = 1, 2, \dots, m$$

Then $\{M(t); t \geq 0\}$ is a time-homogeneous Markov process.

Proof. From Proposition 5.1 we know that $\{M_i; i = 0, 1, 2, \dots\}$ is a time-homogeneous Markov chain. Since $\{M_i; i = 0, 1, 2, \dots\}$ is independent of $\{H(t), t \geq 0\}$ that is a Poisson process $\{H(t), t \geq 0\}$ with parameter λ, and $M(t) = M_{H(t)}$, we conclude that $\{M(t); t \geq 0\}$ is a time homogeneous Markov process (Feller 1971). □

Remarks.

(1) Corollary 5.1 is a theoretical result that the cumulative number of observed failures in conventional random testing obeys a time-homogeneous Markov process. To the best knowledge of the present authors, this has not been exposed in existing literature in a mathematically rigorous manner.
(2) We should note that the above Markov chain is not a Poisson process in general, although it is often assumed in software reliability modeling that the cumulative number of observed failures is Poisson distributed (Xie 1991, Lyu 1996, Cai 1998).

5.2 Case 2: $p_{ij} \equiv p$; $i,j = 1, 2, \dots, m$

Following Section 4, in this case we can obtain the system of backward differential equations for the process as follows

$$\begin{cases} \frac{dP_{\langle c,l\rangle\langle b,j\rangle}(s,t)}{dt} = -\lambda P_{\langle c,l\rangle\langle b,j\rangle}(s,t) + \lambda\sum_{k=1}^{m} P_{\langle c,l\rangle\langle b,k\rangle}(s,t)[1 - N\theta_k]p; \quad 0 \leq b = c < N \\ \frac{dP_{\langle c,l\rangle\langle b,j\rangle}(s,t)}{dt} = -\lambda P_{\langle c,l\rangle\langle b,j\rangle}(s,t) + \lambda\sum_{k=1}^{m} P_{\langle c,l\rangle\langle b,k\rangle}(s,t)[1 - (N-b)\theta_k]p \\ \qquad\qquad +\lambda\sum_{k=1}^{m} P_{\langle c,l\rangle\langle b-1,k\rangle}(s,t)[N - b + 1]\theta_k p; \quad c + 1 \leq b < N \\ \frac{dP_{\langle c,l\rangle\langle b,j\rangle}(s,t)}{dt} = \lambda\sum_{k=1}^{m} P_{\langle c,l\rangle\langle N-1,k\rangle}(s,t)\theta_k p; \quad c < N \end{cases}$$

where $l, j \in \{1, 2, \dots, m\}$. Let us consider the following reasonable initial conditions of the above system of equation:

(1) Suppose an action is finished at time instant s (and thus the next action starts at time instant $s+$). It holds

$$P_{\langle c,l\rangle\langle b,j\rangle}(s,s) = \begin{cases} p_{ij} = p & \text{if } b = c \text{ or } b = c + 1 \\ 0 & \text{otherwise} \end{cases}; \quad l,j = 1, 2, \dots, m$$

(2) Suppose s is not the time instant at which an action is finished. It holds

$$P_{\langle c,l\rangle\langle b,j\rangle}(s,s) = \begin{cases} 1 & \text{if } b = c \text{ and } j = l \\ 0 & \text{otherwise} \end{cases}; \quad l,j = 1, 2, \dots, m$$

We immediately conclude that the above system of equations has a unique solution if either of the above two initial conditions is satisfied.

Let $\sigma\{\langle M(u), A(u)\rangle, u \geq s\}$ and $\sigma\{\langle M(s), A(s)\rangle\}$ be the σ-algebra generated by $\{\langle M(u), A(u)\rangle, u \geq s\}$ and $\{\langle M(s), A(s)\rangle\}$, respectively. Denote $P_{\langle M(s),A(s)\rangle}$ as the probability measure on measurable space (Ω, F), and $E_{\langle M(s),A(s)\rangle}$ as the expectation with respect to $P_{\langle M(s),A(s)\rangle}$. Note that $P_{\langle M(s),A(s)\rangle}$ has the property: for any $\sigma\{\langle M(u), A(u)\rangle, u \geq s\}$ measurable function f,

$$E_{\langle M(s),A(s)\rangle} f = E\{f|\langle M(s), A(s)\rangle\}$$

$$\text{If } f = I_{\langle M(t),A(t)\rangle}(\Lambda) = \begin{cases} 1 & \text{if } \langle M(t), A(t)\rangle \in \Lambda \\ 0 & \text{otherwise} \end{cases}, \text{where } \Lambda \subset \{0, 1, 2, \dots\} \times \{0, 1, 2, \dots\},$$

then

$$E_{\langle M(s),A(s)\rangle} I_{\langle M(t),A(t)\rangle}(\Lambda) = P_{\langle M(s),A(s)\rangle}(\langle M(t), A(t)\rangle \in \Lambda) = \Pr\{\langle M(t), A(t)\rangle \in \Lambda | \langle M(s), A(s)\rangle\}$$

PROPOSITION 5.2. Suppose an action is finished at the time instant s with state $\langle M(s), A(s)\rangle = \langle c, l\rangle$ or $\langle M(s), A(s)\rangle = \langle c, l1\rangle$, where $l \neq l1$ and $l, l1 \in \{1, 2, \dots, m\}$.

Then

$$P_{\langle c,l\rangle} = P_{\langle c,l1\rangle}$$

Proof. See Appendix. □

Let

$$t_0 = 0$$

$$t_i = \inf\{t > t_{i-1} | M(t) \neq M(t_{i-1})\}$$

$$\tau_i = t_i - t_{i-1}; \quad i = 1, 2, \cdots, N$$

That is, τ_1 denotes the time to the first observed failure, and τ_i denotes the times between the ith observed failure and the $(i-1)$th failure. We have the following proposition.

PROPOSITION 5.3. Suppose that it holds

$$p_{ij} \equiv p; \quad i, j = 1, 2, \dots, m$$

Then τ_1 is exponentially distributed with a random parameter.

Proof. From Proposition 5.2 we have,

$$\begin{aligned}
P_{\langle M(0),A(0+)\rangle}\{\tau_1 > t + s\} &= P_{\langle M(0),A(0+)\rangle}\{\tau_1 > t + s, \tau_1 > s\} \\
&= P_{\langle M(0),A(0+)\rangle}\{\tau_1 \circ \vartheta_s + s > t + s, \tau_1 > s\} \\
&= P_{\langle M(0),A(0+)\rangle}\{\tau_1 \circ \vartheta_s > t, \tau_1 > s\} \\
&= E_{\langle M(0),A(0+)\rangle}[P_{\langle M(s),A(s+)\rangle}\{\tau_1 > t\} I_{\{\tau_1 > s\}}] \\
&= \sum_{k=1}^{m} E_{\langle M(0),A(0+)\rangle}\left[P_{\langle M(s),A(s+)\rangle}\{\tau_1 > t\} I_{\{\tau_1 > s, A(s+)=k\}}\right] \\
&= \sum_{k=1}^{m} E_{\langle M(0),A(0+)\rangle}\left[P_{\{M(0),k\}}\{\tau_1 > t\} I_{\{\tau_1 > s, A(s+)=k\}}\right] \\
&= P_{\langle M(0),A(0+)\rangle}\{\tau_1 > t\} \sum_{k=1}^{m} P_{\langle M(0),A(0+)\rangle}\{\tau_1 > s, A(s+) = k\} \\
&= P_{\langle M(0),A(0+)\rangle}\{\tau_1 > t\} P_{\langle M(0),A(0+)\rangle}\{\tau_1 > s\}
\end{aligned}$$

This implies

$$\Pr\{\tau_1 > t | M(0) = 0, A(0+) = l\} = \mathrm{e}^{-\kappa(0,l)t}$$

where $\kappa(0, l)$ is a constant depending on the initial state of the process $\{\langle M(t), A(t)\rangle; t \geq 0\}$ at time instant $t = 0$. Note $A(0+)$ is a random variable. We can reasonably say that the time to the first observed failure is exponentially distributed with a random parameter depending on the choice of the first action A_1. □

Remarks.

(1) In the proof of Proposition 5.3 ϑ_s denotes the shift operator defined as follows

$$\vartheta_s(\langle M(t), A(t)\rangle) = \langle M(t+s), A(t+s)\rangle$$

and

$$\tau_1 \circ \vartheta_s = \inf\{t > 0, M(t) \neq 0\} \circ \vartheta_s = \inf\{t > 0, M(t+s) \neq 0\}$$

(2) From the Markovian property of $\{\langle M(t), A(t)\rangle; t \geq 0\}$ and the proof of Proposition 5.3, we conclude that

$$P_{\langle M(\tau_i), A(\tau_i+)\rangle}(\tau_{i+1} > t) = \mathrm{e}^{-\kappa(M(\tau_i), A(\tau_i+))t}$$

5.3 Case 3: $\theta_i \equiv \theta$; $i = 1, 2, \ldots, m$

This case can be obtained by setting $\theta_i \equiv \theta; \quad i = 1, 2, \ldots, m$ in Case 1 directly. In this way all the testing actions are equal in terms of the defect detection rates and there is no need to distinguish them. Alternatively, it can also be obtained by setting

$$p_{ij} = \begin{cases} 1 & \text{if } j = j_0, \forall i \\ 0 & \text{otherwise} \end{cases}$$

That is, during software testing only action j_0 is taken and no other distinct actions are taken. This is equivalent to $\theta_i \equiv \theta_{j_0}; \quad i = 1, 2, \ldots, m$.

PROPOSITION 5.4. Suppose it holds

$$\theta_i \equiv \theta; \quad i = 1, 2, \ldots, m$$

Then the Markov process $\{M(t); t \geq 0\}$ is not an independent increment process.

Proof. See Appendix 5.4. □

PROPOSITION 5.5. Suppose it holds

$$\theta_i \equiv \theta; \quad i = 1, 2, \ldots, m$$

$\tau_1, \tau_2, \ldots, \tau_{N-1}$ are exponentially distributed as follows

$$\Pr\{\tau_1 > t | M(0)\} = \mathrm{e}^{-N\theta\lambda t}$$

$$\Pr\{\tau_i > t | M(0)\} = \mathrm{e}^{-(N-i)\theta\lambda t}; \quad i = 2, 3, \ldots, N-1$$

Proof. See Appendix 5.5. □

Up to this point we can see that $\{M_i, i = 1, 2, \ldots\}$ is not a Markov chain and $\{M(t); t \geq 0\}$ is not a Markov chain process in general. The times between successive observed failures may not be exponentially distributed in general. However if testing actions are independent of testing history and are selected in accordance with a probability distribution as specified in conventional random testing strategy, then $\{M_i, i = 1, 2, \ldots\}$ is a time-homogeneous Markov chain and $\{M(t); t \geq 0\}$ is a time-homogeneous Markov chain process. In general, the time to the first observed failure is exponentially distributed with respect to the probability measure $P_{M(0)}$‡, but this does not guarantee that the time between successive observed failures are exponentially distributed with respect to the probability measure $P_{\langle M(0), A(0+)\rangle}$. If the action probability distribution is a uniform distribution, i.e. $p_{ij} \equiv p; \quad i, j = 1, 2, \ldots, m$, then the times between successive observed failures are exponentially distributed with a random parameter with respect to the probability measure $P_{\langle M(0), A(0+)\rangle}$. If the defect detection rates of various testing actions can further be identical or during testing only a specific action testing is taken, then the times between successive observed failures are exponentially distributed with a non-random parameter. The results presented so far give a mathematically rigorous characterization of software reliability growth behavior under various random testing strategies including Markov usage model based testing and conventional random testing. Although it is commonly assumed that the times between successive observed failures are exponentially distributed (the exponentiality assumption) and the cumulative number of observed failures follows a non-homogeneous Poisson process (the NHPP assumption), little theoretical justifications are available to justify or refute them and they are seldom related to testing strategies. We can conclude that the exponentiality assumption can only hold under some specific cases. Further even for the special case $\theta_i \equiv \theta; \quad i = 1, 2, \ldots, m$, Proposition 5.4 shows that the corresponding $\{M(t); t \geq 0\}$ is not an independent increment process and thus is not a NHPP either. We arrive at that the NHPP assumption adopted in software reliability modeling is theoretically false. We have shown in a separate paper (Cai *et al.* 2004) that the NHPP assumption is empirically false.

6. Software testing stability

From Section 5, we see that in some special cases $\{M(t); t \geq 0\}$ is a Markov process and τ_1 can even be exponentially distributed. This is a desirable property. However Proposition 3.1 states that $\{M(t); t \geq 0\}$ is not a Markov process in general. A natural question is that how much $\{M(t); t \geq 0\}$ deviates from a Markov process in general, or whether the deviations can

‡$P_{M(0)}$ is the probability measure of the time-homogeneous Markov process $\{M(t); t \geq 0\}$ with the initial state $M(0)$. Refer to Proposition 8.3.

eventually diminish as software testing proceeds. If the deviations eventually diminish, then we can say that the software testing process approaches a desirable property and is thus stable in some sense. To clarify this, let us present some of theoretical observations first and then discuss the potentials of introducing a notion of stability for software testing.

6.1 Theoretical observations

Consider how the effects of variations of $\{\theta_1, \theta_2, \ldots, \theta_m\}$ may diminish as software testing proceeds. Suppose we have $\theta_1 > \theta_2 > \cdots > \theta_m$. Further, software testing reveals the ith failure at time instant t_i with $0 = t_0 < t_1 < t_2 < \cdots < t_{N-1} < t_N$. Then according to the assumptions presented in Section 2.2, the number of defects remaining in the software under test during the time interval (t_{i-1}, t_i) is $l_i = N - i + 1$. The corresponding defect detection rate of the software under test is $\alpha_i = l_i\theta_j$ if an action corresponding to C_j is taken during the time interval. Note that we can have

$$l_i\theta_j = l_i\theta_m + l_i(\theta_j - \theta_m) = l_i\theta_m + \varepsilon_{ij}$$

where $\varepsilon_{ij} = l_i(\theta_j - \theta_m)$. The term $l_i\theta_m$ corresponds to a virtual software testing process with $\theta_1 = \theta_2 = \cdots = \theta_m$, whose $\{M(t); t \geq 0\}$ is a Markov process according to Proposition 5.3. The term ε_{ij} can be interpreted as a perturbation attached to the virtual process and introduced by action j. Since any of the m distinct actions can be taken during the time interval (t_{i-1}, t_i), ε_{ij} varies from $l_i(\theta_m - \theta_m)$ to $l_i(\theta_1 - \theta_m)$. Denote

$$\varepsilon^{(i)} = l_i(\theta_1 - \theta_m)$$

Then

$$\varepsilon_{ij} \leq l_i(\theta_1 - \theta_m) = \varepsilon^{(i)}; \quad j = 1, 2, \ldots, m$$

Note that l_i decreases monotonically as software testing proceeds and $\varepsilon^{(i)}$ eventually approaches zero, we may expect that $\{M(t); t \geq 0\}$ of the actual software testing process, which is viewed as the composition of a virtual process and perturbation, will approach a Markov process. In other words, the software testing process demonstrates stability of some kind for any given parameters $\{\theta_1, \theta_2, \ldots, \theta_m\}$. More rigorously, let $\bar{Q}$ be the Q-matrix determined by the process $\langle \bar{M}(t), \bar{A}(t); t \geq 0\rangle$ with $\{\bar{\theta}_1, \bar{\theta}_2, \ldots, \bar{\theta}_m\}$, $\bar{\theta}_1 = \bar{\theta}_2 = \cdots = \bar{\theta}_m = \theta \in (0, 1), N\theta < 1$, the test transition probability matrix $\lfloor p_{ij} \rfloor$, and the initial action distribution $\{p_1, p_2, \ldots, p_m\}$. Let $\bar{P}$ be the corresponding probability measure. We have the following propositions.

PROPOSITION 6.1. For an arbitrary bounded function $f : \{0, 1, \ldots, N\} \times \{1, 2, \ldots, m\} \to \{-\infty, \infty\}$, it holds

$$\sup_{u \geq t} \left| E_{\langle M(0), A(0+)\rangle} f(\langle M(u), A(u)\rangle) - E_{\langle \bar{M}(0), \bar{A}(0+)\rangle} f(\langle \bar{M}(u), \bar{A}(u)\rangle) \right| \leq \gamma e^{-t}; \quad \forall t > 0$$

with

$$\gamma = 2 \max_{x \in \{0,1,\ldots,N\} \times \{1,2,\ldots,m\}} |f(x)| \times E_{\langle M(0), A(0+)\rangle} e^{\tau} \times E_{\langle \bar{M}(0), \bar{A}(0+)\rangle} e^{\bar{\tau}}$$

$$\tau = \inf\{t > 0, \langle M(t), A(t)\rangle \neq \langle M(t-), A(t+)\rangle\}$$

$$\bar{\tau} = \inf\{t > 0, \langle \bar{M}(t), \bar{A}(t)\rangle \neq \langle \bar{M}(t-), \bar{A}(t+)\rangle\}$$

Proof. Let $P = P \times \bar{P}$ be the independent product probability of P and $\bar{P}$. From the coupling formula of Markov processes (Chen 2003), it holds

$$\begin{aligned}&\left|E_{\langle M(0),A(0+)\rangle} f(\langle M(t), A(t)\rangle) - E_{\langle \bar{M}(0),\bar{A}(0+)\rangle} f(\langle \bar{M}(t), \bar{A}(t)\rangle)\right| \\ &\quad \leq 2 \max_{x\in\{0,1,\ldots,N\}\times\{1,2,\ldots,m\}} |f(x)| \times P_{\langle M(0),A(0+);\bar{M}(0),\bar{A}(0+)\rangle}\{T > t\}\end{aligned}$$

where $T = \inf\{t > 0, \langle M(t), A(t); \bar{M}(t), \bar{A}(t)\rangle \neq \langle M(t-), A(t+); \bar{M}(t-), \bar{A}(t+)\rangle\}$.

Note that $T = \min\{\tau, \bar{\tau}\}$. Then

$$\begin{aligned}P_{\langle M(0),A(0+);\bar{M}(0),\bar{A}(0+)\rangle}\{T > t\} &= P_{\langle M(0),A(0+);\bar{M}(0),\bar{A}(0+)\rangle}\{\min\{\tau, \bar{\tau}\} > t\} \\ &\leq E_{\langle M(0),A(0+);\bar{M}(0),\bar{A}(0+)\rangle}\left[\mathrm{e}^{-t} \times \mathrm{e}^{\min\{\tau,\bar{\tau}\}}\right] \\ &\leq E_{\langle M(0),A(0+);\bar{M}(0),\bar{A}(0+)\rangle}\left[\mathrm{e}^{-t} \times \mathrm{e}^{\tau} \times \mathrm{e}^{\bar{\tau}}\right] \\ &= \mathrm{e}^{-t} E_{\langle M(0),A(0+)\rangle}\mathrm{e}^{\tau} E_{\langle \bar{M}(0),\bar{A}(0+)\rangle}\mathrm{e}^{\bar{\tau}}\end{aligned}$$

where E denotes the mathematical expectation corresponding to P. This completes the Proof. □

PROPOSITION 6.2. For an arbitrary bounded function $f : \{0, 1, \ldots, N\} \to \{-\infty, \infty\}$, it holds

$$\sup_{u\geq t} \left|E_{\langle M(0),A(0+)\rangle} f(M(u)) - E_{\bar{M}(0)} f(\bar{M}(u))\right| \leq \gamma \mathrm{e}^{-t}; \quad \forall t > 0$$

with

$$\gamma = 2\max_{x\in\{0,1,\ldots,N\}} |f(x)| \times E_{\langle M(0),A(0+)\rangle}\mathrm{e}^{\tau} \times E_{\bar{M}(0)}\mathrm{e}^{\bar{\tau}}$$

$$\tau = \inf\{t > 0, \langle M(t), A(t)\rangle \neq \langle M(t-), A(t+)\rangle\}$$

$$\bar{\tau} = \inf\{t > 0, \bar{M}(t) \neq \bar{M}(t-)\}$$

Proof. The corollary can be proved in a similar manner as that of Proposition 6.1. □

Remarks.

(1) Let

$$f(\langle M(t), A(t)\rangle) = \begin{cases} 1 & \text{if } M(t) = b \\ 0 & \text{otherwise} \end{cases},$$

where $b \in \{0, 1, \ldots, N\}$. Then we have

$$\sup_{u\geq t} \left|P_{\langle M(0),A(0+)\rangle}\{M(u) = b\} - P_{\bar{M}(0)}\{\bar{M}(u) = b\}\right| \leq \gamma \mathrm{e}^{-t}$$

(2) Propositions 6.1 and 6.2 and the above remark roughly indicate that as software testing proceeds, the difference between $\{M(t); t \geq 0\}$ and a Markov process diminishes in an exponential manner.

6.2 Discussion

Propositions 6.1 and 6.2 present useful results for us to understand the stability behavior of software testing. Stability is usually concerned with if a desirable property of a system or process can resume under an undesirable environment as the system or process evolves without extra influence. For example, Lyapunov' notion of stability is mainly concerned with if a dynamic system can approach an equilibrium state from an initial state other than the equilibrium state (Khalil 2001). Here the desirable property is that the system can remain in or is close to the given equilibrium state, and the undesirable environment refers to the condition that the system is not in the equilibrium state at the beginning. Dijkstra's notion of self-stability is concerned with if a system or a distributed computing algorithm can automatically resume normal operation following the occurrence of (transient) faults (Dolev 2000). Here the desirable property is that the system keeps normal operation, and the undesirable environment is that faults may occur to the system. The notion of stochastic stability characterizes the desirable property against the undesirable environments in the context of probability measure (Kushner 1967). A manufacturing process that delivers mass products may be considered as stable if the mean and variance of some of measures of the products tend to be time-invariant after some time. Here the desirable property is that the delivered products are independent of the manufacturing time in terms of the mean and variance of the given measures. Similarly for the software testing process, suppose that the desirable property is determined by the process $\langle \bar{M}(t), \bar{A}(t); t \geq 0 \rangle$ and the undesirable environment is that $\theta_1 = \theta_2 = \cdots = \theta_m$ does not hold. Then Propositions 6.1 and 6.2 indicate that the software testing process is stable.

There are many desirable properties that we want the software testing process to have. These desirable properties can be theoretical and/or practical. The desirable property determined by the Markov process $\langle \bar{M}(t), \bar{A}(t); t \geq 0 \rangle$ is theoretical. The desirable property that the times between successive observed failures are exponentially distributed is also theoretical. We can identify more theoretical desirable properties for the software testing process as will be observed in Section 8. These theoretical properties help us to understand the stability behavior of software testing from a theoretical perspective. An interesting question is that how the remaining time to software testing stopping behave as software testing proceeds.

In practice the desirable properties may be dependent on the given testing goals. As addressed in Section 1, there are various testing goals. For the goal of software reliability improvement, one desirable property may be that as many defects as possible can be detected and removed within a given number of tests. A second desirable property may be that a given number of defects, if any, are detected and removed with the least number of tests. For the goal of software reliability assessment, one desirable property may be that the resulting reliability estimate is the most trustworthy in the sense that the variance of the corresponding reliability estimator is minimized. A second desirable property may be that the variance of the software estimate decreases to a given threshold with the least number of tests. Similar desirable properties can be identified for the software testing process if the testing goal is for test suite assessment.

The importance of the software testing stability problem stems from the fact that undesirable environments are universally present in the software testing process and may have significant effects on the software testing process. An obvious undesirable environment is that the requirement that $\theta_1 = \theta_2 = \cdots = \theta_m$ can hardly be valid in practice. Recall that the

software testing process is mathematically determined by $\{\theta_1, \theta_2, \ldots, \theta_m\}$, $\lfloor p_{ij} \rfloor_{m\times m}$ and $\{\Pr\{A_1 = j\}, j = 1, 2, \ldots, m\}$. Unfortunately, these parameters can not be determined or estimated accurately in reality. For example, there are several methods for determining $\lfloor p_{ij} \rfloor_{m\times m}$ (Walton 1995, Semmel and Linton 1998, Gutjahr 2000, Walton and Poore 2000, Poore *et al.* 2000), but it is difficult to assign accurate and optimal values to them. Further, $\{\theta_1, \theta_2, \ldots, \theta_m\}$ also suffers from variations. Even for a single piece of software under test, different test suite may lead to different values of $\{\theta_1, \theta_2, \ldots, \theta_m\}$. For a single testing strategy applied to different software, the corresponding values of $\{\theta_1, \theta_2, \ldots, \theta_m\}$ can hardly be the same. A natural question is whether various software testing processes with different parameters can demonstrate invariant property or stability of some kind. More specifically, given nominal or theoretical values for the parameters $\{\theta_1, \theta_2, \ldots, \theta_m\}$, $\lfloor p_{ij} \rfloor_{m\times m}$ and $\{\Pr\{A_1 = j\}, j = 1, 2, \ldots, m\}$, how does the underlying software testing process behave against possible variations attached to the nominal values.

The desirable property that as many defects as possible are be detected and removed within a given number of tests may be achieved by a specific testing process or testing strategy for given software under test. However the desirable property may become unachievable for the testing process or testing strategy for another software system under test since two software systems can be extremely different. There are scientific software, reactive software, embedded software, and so on. It is important to investigate what desirable properties can be achievable under what conditions (under what desirable environments) and how the desirable properties can be achieved. One more undesirable environment is due to the software development schedule which is usually stringent. Software must be released or deployed in due course and it is also important to investigate how the undesirable schedule constraints affect the desirable properties.

Although we have not presented a formal definition for software testing stability, we can roughly identify two categories of stability for the software testing process: inner stability and outer stability. Let Γ denote the testing process determined by the three sets of parameters $\{\theta_1, \theta_2, \ldots, \theta_m\}$, $\lfloor p_{ij} \rfloor_{m\times m}$ and $\{\Pr\{A_1 = j\}, j = 1, 2, \ldots, m\}$. The inner stability of software testing examines that for any given sets of parameters, if the corresponding testing process eventually demonstrates an invariant property or pattern as software testing proceeds. This invariant property should be irrespective of the given sets of parameters. More specifically, suppose there is a desirable testing process $\bar{\Gamma}$ that demonstrates some parameter independent property. The inner stability of software testing examines if Γ approaches $\bar{\Gamma}$ in some sense as software testing proceeds.

On the other hand, suppose that nominal values are given for the three sets of parameters $\{\theta_1, \theta_2, \ldots, \theta_m\}$, $\lfloor p_{ij} \rfloor_{m\times m}$ and $\{p_j, j = 1, 2, \ldots, m\}$. Denote the corresponding testing process by Γ. The outer stability of software testing is concerned with the effects of perturbations to the nominal values on the behavior of the software testing process. Let

$$\theta_j^{\Delta} = \theta_j + \Delta\theta_j$$

$$p_{ij}^{\Delta} = p_{ij} + \Delta p_{ij}$$

$$p_j^{\Delta} = p_j + \Delta p_j; \quad i, j = 1, 2, \ldots, m$$

Denote the testing process determined by $\{\theta_1^{\Delta}, \theta_2^{\Delta}, \ldots, \theta_m^{\Delta}\}$, $\lfloor p_{ij}^{\Delta} \rfloor_{m\times m}$ and $\{p_j^{\Delta}, j = 1, 2, \ldots, m\}$ by Γ^{Δ}. The outer stability of software testing examines if Γ^{Δ} approaches Γ in some sense as software testing proceeds. Here we note that $E_{\langle M(0), A(0+)\rangle} f(\langle M(t), A(t)\rangle)$ is a continuous function of $\{\theta_1, \theta_2, \ldots, \theta_m\}$, $\lfloor p_{ij} \rfloor_{m\times m}$ and $\{p_j, j = 1, 2, \ldots, m\}$.

7. Behavior of $\{EM_i, i = 1, 2, \dots\}$ and $EM(t)$

In this section, we examine the behavior of $\{EM_i, i = 1, 2, \dots\}$, or the expected behavior of $\{M_i, i = 1, 2, \dots\}$ as software testing proceeds. Recall that M_i denotes the accumulative number of failures revealed in the first i tests or actions. We have the following Proposition for the special case $p_{ij} \equiv p_j$; $i,j = 1, 2, \dots, m$.

PROPOSITION 7.1. Under the model assumptions presented in Section 2.2 with $p_{lj} \equiv p_j$; $l,j = 1, 2, \dots, m$, it holds

$$EM_i = N[1 - \mathrm{e}^{-i\beta}]; \quad i = 1, 2, \dots$$

where

$$\beta = \ln \frac{1}{1 - \hat{\theta}}$$

$$\hat{\theta} = \sum_{j=1}^{m} p_j \theta_j$$

Proof. See Appendix. □

PROPOSITION 7.2. Under the model assumptions presented in Section 2.2 with $p_{lj} \equiv p_j$; $l,j = 1, 2, \dots, m$, it holds

$$EM(t) = N[1 - \mathrm{e}^{-\lambda \hat{\theta} t}]; \quad t \geq 0$$

where

$$\hat{\theta} = \sum_{j=1}^{m} p_j \theta_j$$

Proof. We have

$$\begin{aligned}
EM(t) &= \sum_{k=0}^{N} k \Pr\{M(t) = k\} = \sum_{k=0}^{N} k \sum_{i=0}^{\infty} \Pr\{M(t) = k, H(t) = i\} \\
&= \sum_{k=0}^{N} k \sum_{i=0}^{\infty} \Pr\{M(t) = k | H(t) = i\} \Pr\{H(t) = i\} \\
&= \sum_{k=0}^{N} k \sum_{i=0}^{\infty} \Pr\{M_i = k\} \Pr\{H(t) = i\} \\
&= \sum_{k=0}^{N} k \sum_{i=0}^{\infty} \Pr\{M_i = k\} \frac{(\lambda t)^i}{i!} \mathrm{e}^{-\lambda t} \\
&= \sum_{i=0}^{\infty} \frac{(\lambda t)^i}{i!} \mathrm{e}^{-\lambda t} \sum_{k=0}^{N} k \Pr\{M_i = k\} = \sum_{i=0}^{\infty} \frac{(\lambda t)^i}{i!} \mathrm{e}^{-\lambda t} EM_i
\end{aligned}$$

From Proposition 7.1 we obtain

$$EM(t) = \sum_{i=0}^{\infty} \frac{(\lambda t)^i}{i!} e^{-\lambda t} N\left[1 - (1 - \hat{\theta})^i\right] = N e^{-\lambda t}\left[e^{\lambda t} - e^{\lambda(1-\hat{\theta})t}\right] = N\left[1 - e^{-\lambda \hat{\theta} t}\right]$$

□

Propositions 7.1 and 7.2 show that under conventional random testing strategy, the expected behavior of M_i or $M(t)$ approaches N monotonically in an exponential manner. The growth rate of EM_i or $EM(t)$ is determined by the average defect detection rate $\hat{\theta}$ or/and the testing intensity λ. $\hat{\theta}$ and λ can be treated as a characteristic measure of conventional random testing strategy in some sense. This raises an interesting question: are there any invariant measures for a given testing strategy that behave as the eigenvalues of a given matrix? This should be a topic deserving future investigation.

However, it looks difficult to mathematically investigate the behavior of $EM(t)$ for general cases. Rather, the following simulation example may help us to understand the behavior of $EM(t)$ in general.

Example 7.1. Let $N = 30$, $m = 3$. That is, there are 30 defects remaining in the software under test at the beginning of software testing and we have three distinct testing actions. $\theta_1 = 0.00345$, $\theta_2 = 0.00745$, and $\theta_3 = 0.00145$. The first action is selected in accordance with the probability distribution {0.51290893578572, 0.22429752010530, 0.26279354410898}. The transition probabilities of the Markov chain are as follows:

0.35039537369758	0.06174214927836	0.58786247702406
0.43367104392204	0.40166042914365	0.16466852693431
0.11596823256275	0.06902933957153	0.81500242786572

Further, the parameter of the Poisson process is $\lambda = 5.0$. We do simulation for 100 times and obtain 100 realizations of $M(t)$. The behavior of $EM(t)$ can be obtained accordingly by averaging the 100 realizations of $M(t)$. Figure 2 shows the behavior of $EM(t)$ up to $t = 500$. $EM(t)$ looks to fit an exponential law. This can be justified by figure 3, where

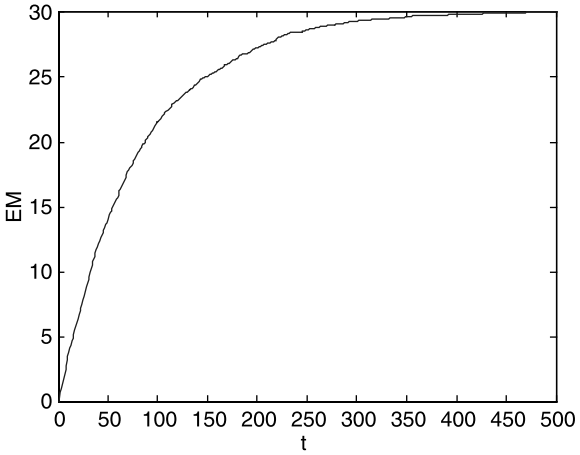

Figure 2. Simulation results of $EM(t)$ for example 7.1.

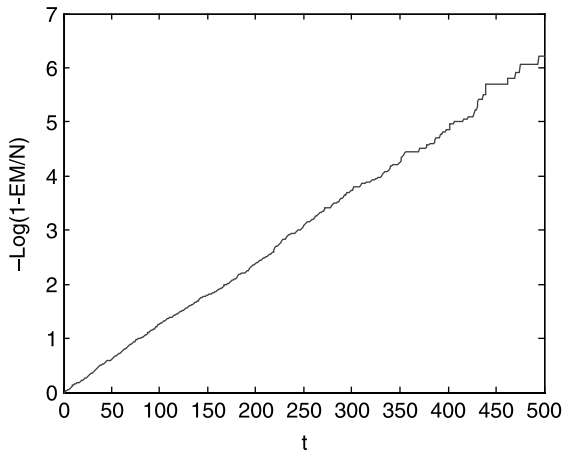

Figure 3. Simulation results of $-\ln(1-(EM(t)/N))$ for example 7.1.

$-\ln(1-EM(t)/N)$ looks like a linear function of t. Note that the parameter setting for simulation is chosen rather arbitrarily since it is not sure how to choose the required parameters otherwise.

PROPOSITION 7.3. Under the model assumptions presented in Section 2.2, it holds

$$\left|E_{\langle M(0),A(0+)\rangle}M(t) - E_{\bar{M}(0)}\bar{M}(t)\right| \leq \gamma N \mathrm{e}^{-t}$$

where $\{\bar{M}(t), t \geq 0\}$ is the Markov process corresponding to the case $\bar{\theta}_1 = \bar{\theta}_2 = \cdots = \bar{\theta}_m = \theta \in (0,1), N\theta < 1$ as identified in Section 6.2, and γ is defined in Proposition 6.1.

Proof. For any measurable function f defined on $\{0, 1, 2, \ldots\}$, from Proposition 6.1 we have

$$\sup_{u \geq t > 0} \left|E_{\langle M(0),A(0+)\rangle} f(M(t)) - E_{\bar{M}(0)} f(\bar{M}(t))\right| \leq \gamma \mathrm{e}^{-t} \max_{1 \leq k \leq N} f(k)$$

Let $f(x) = x$, we arrive at

$$\left|E_{\langle M(0),A(0+)\rangle}M(t) - E_{\bar{M}(0)}\bar{M}(t)\right| \leq \gamma N \mathrm{e}^{-t}$$

□

Remarks.

(1) Although $\{M(t), t \geq 0\}$ is not a Markov process in general and the corresponding behavior of $EM(t)$ does not strictly fit an exponential law, Proposition 7.3 indicates that $M(t)$ is expected to approach an exponential law asymptotically as software testing proceeds. Example 7.1 shows that $EM(t)$ is close to an exponential law.

(2) Obviously, it holds

$$\lim_{t \to \infty} \frac{E_{\langle M(0),\, A(0+)\rangle}M(t)}{N\left(1 - \mathrm{e}^{-\lambda \hat{\theta} t}\right)} = 1$$

8. Behavior of times between successive distinctions

In Section 5 we examine the behavior of times between successive observed failures and show that they are exponentially distributed for some special cases. However we have not examined if they are independent. On the other hand, we implicitly assume we the time instants of interest in the software testing process are those at which failures are revealed. These time instants can be identified in terms of $\{M(t); t \geq 0\}$. However the time instants of interest in the software testing process can also be those at which a distinct action is taken. These time instants can be identified in terms of $\{A(t); t \geq 0\}$. One may also be interested in the time instants that can be identified in terms of $\langle M(t), A(t); t \geq 0\rangle$. We collectively call various time instants of interest in the software testing process *distinction*. In this section we go beyond Section 5 and examine the behavior of times between successive distinctions for special cases as well as for general cases.

8.1 The discrete time domain

In this subsection the time instants of interest are identified in terms of $\{M_i; i = 1, 2, \ldots\}$. That is, the times between successive distinctions are discrete. Let

$$t_0 = 0$$
$$t_i = \inf\{j | M_j = i\}$$
$$\tau_i = t_i - t_{i-1}; \quad i = 1, 2, \ldots, N$$

The question of interest is if $\tau_1, \tau_2, \ldots, \tau_N$ are independent. Unfortunately, the answer is negative in general. Let us consider an example for τ_1 and τ_2.

First we have

$$\Pr\{\tau_2 = 1 | \tau_1 = k\} = \frac{\Pr\{M_{k+1} = 2, M_k = 1, M_{k-1} = \cdots = M_1 = 0\}}{\Pr\{M_k = 1, M_{k-1} = \cdots = M_1 = 0\}}$$

We note that Section 3 has shown how to calculate $\Pr\{M_{i+1}, M_i, \ldots, M_1, M_0\}$ directly and thus the above formula can be used to verify if τ_1 and τ_2 are independent.

Example 8.1. Let $N = 30$, $m = 2$, $\theta_1 = 0.013567$, $\theta_2 = 0.007133$, $p_{11} = 0.607815$, $p_{12} = 0.392185$, $p_{21} = 0.525566$, $p_{22} = 0.474434$, $p_1 = 0.706349$, $p_2 = 0.293651$. Then we have

$$\Pr\{\tau_2 = 1 | \tau_1 = 1\} = 0.317512$$
$$\Pr\{\tau_2 = 1 | \tau_1 = 2\} = 0.316016$$
$$\Pr\{\tau_2 = 1 | \tau_1 = 3\} = 0.315874$$
$$\Pr\{\tau_2 = 1 | \tau_1 = 4\} = 0.315862$$

This implies that τ_1 and τ_2 are not independent.

PROPOSITION 8.1. $\tau_1, \tau_2, \ldots, \tau_N$ in the context of the discrete time domain are not independent in general.

Proof. See the above example □

8.2 The continuous time domain

In this subsection we identify the distinctions in terms of $\langle M(t), A(t); t \geq 0\rangle$. Let

$$\chi_0 = 0$$
$$\chi_i = \inf\{t > \chi_{i-1} | \langle M(t), A(t)\rangle \neq \langle M(\chi_{i-1}), A(\chi_{i-1}+)\rangle\}$$
$$\eta_i = \chi_i - \chi_{i-1}; \quad i = 1, 2, \ldots, N$$

PROPOSITION 8.2. Let

$$k_{\max}(u) = \arg \max_{1\leq k\leq m} P_{\langle 1,k\rangle}\{\eta_1 < u\}$$
$$k_{\min}(u) = \arg \min_{1\leq k\leq m} P_{\langle 1,k\rangle}\{\eta_1 < u\}$$

Suppose $k_{\max}(u) \neq k_{\min}(u)$ holds for some $u \in [0, \infty)$. Then for any $i, j = 1, 2, \ldots, N$, $i \neq j$, η_i and η_j are not independent.

Proof. We only need to consider the case of $i = 1$ and $j = 2$. For any measurable functions f and g defined on $[0, \infty)$, we have

$$\begin{aligned}
E_{\langle M(0),A(0+)\rangle}[f(\eta_2)g(\eta_1)] &= E_{\langle M(0),A(0+)\rangle}\left[f\left(\eta_1 \circ \vartheta_{\chi_1}\right)g(\eta_1)\right] \\
&= E_{\langle M(0),A(0+)\rangle}\left[g(\eta_1)E_{\langle M(\chi_1),A(\chi_1+)\rangle}f(\eta_1)\right] \\
&= E_{\langle M(0),A(0+)\rangle}\left[g(\eta_1)E_{\langle 1,A(\chi_1+)\rangle}f(\eta_1)\right] \\
&= \sum_{k=1}^{m} E_{\langle M(0),A(0+)\rangle}\left[g(\eta_1)E_{\langle 1,k\rangle}[f(\eta_1)] \times I_{\{A(\chi_1+)=k\}}\right] \\
&= \sum_{k=1}^{m} \{E_{\langle 1,k\rangle}[f(\eta_1)]\}E_{\langle M(0),A(0+)\rangle}\left[g(\eta_1) \times I_{\{A(\chi_1+)=k\}}\right]
\end{aligned}$$

This implies for any $u_1, u_2 \in [0, \infty)$

$$P_{\langle M(0),A(0+)\rangle}\{\eta_2 < u_2, \eta_1 < u_1\} = \sum_{k=1}^{m} P_{\langle 1,k\rangle}\{\eta_1 < u_1\}P_{\langle M(0),A(0+)\rangle}\{\eta_1 < u_1, A(\chi_1+) = k\}$$

Since $k_{\max}(u) \neq k_{\min}(u)$ for some $u \in [0, \infty)$, we have

$$P_{\langle M(0),A(0+)\rangle}\{\eta_2 < u, \eta_1 < u\} < P_{\langle 1,k_{\max}(u_2)\rangle}\{\eta_1 < u\}\sum_{k=1}^{m} P_{\langle M(0),A(0+)\rangle}\{\eta_1 < u, A(\chi_1+) = k\}$$

$$= P_{\langle 1,k_{\max}(u_2)\rangle}\{\eta_1 < u\}P_{\langle M(0),A(0+)\rangle}\{\eta_1 < u\}$$

□

Remarks.

(1) The condition specified in Proposition 8.2 can be satisfied if there exists a pair of actions k and l such that $\theta_k \neq \theta_l$. This is because if $k_{\max}(u) = k_{\min}(u)$ for all $u \in [0, \infty)$, then for

any $k, l \in \{1, 2, \ldots, m\}$, we have

$$P_{\langle 1,k\rangle}\{\eta_1 < u\} \equiv P_{\langle 1,l\rangle}\{\eta_1 < u\}$$

This implies that

$$E_{\langle 1,k\rangle}e^{x\eta_1} = E_{\langle 1,l\rangle}e^{x\eta_1}; \quad \forall x \leq 0$$

Therefore

$$P_{\langle 1,k\rangle} = P_{\langle 1,l\rangle}$$

However this is impossible for $\theta_k \neq \theta_l$.

(2) The above proposition implies that $\eta_1, \eta_2, \ldots, \eta_N$ are not independent in general.

8.3 Special case

In this subsection we consider the special case approached in Section 5.3 and identify the distinctions in terms of $\{M(t); t \geq 0\}$ as in Section 5. Let

$$t_0 = 0$$

$$t_i = \inf\{t > t_{i-1} | M(t) \neq M(t_{i-1})\}$$

$$\tau_i = t_i - t_{i-1}; \quad i = 1, 2, \ldots, N$$

PROPOSITION 8.3. For the special case approached in Section 5.3, it holds that $\tau_1, \tau_2, \ldots, \tau_N$ are independent with respect to $P_{M(0)}$, which is the probability measure of the time-homogeneous Markov process $\{M(t); t \geq 0\}$ with the initial state $M(0)$.

Proof. See Appendix. □

Remarks.

(1) Proposition 8.1 indicates that $\tau_1, \tau_2, \ldots, \tau_N$ are not independent in general. This implies that the assumption that $\tau_1, \tau_2, \ldots, \tau_N$ are independent, which is widely adopted in software reliability modeling for the sake of mathematical tractability (Xie 1991, Lyu 1996, Cai 1998), is theoretically false in general. However Proposition 8.3 indicates that the independence assumption may hold for some special case.

(2) Since $\tau_1, \tau_2, \ldots, \tau_N$ are independent, we have

$$E_{M(0)}[\tau_i \tau_{i+j}] = \frac{1}{[N - (i-1)][N - (i+j-1)](\lambda\theta)^2}$$

8.4 General cases

For general cases other than the one approached in Section 5.3, it is not easy to obtain accurate quantitative results for various distinctions. However the corresponding software

testing processes may demonstrate some invariant patterns that are related to the inner stability of software testing discussed in Section 6.

8.4.1 Behavior of the times between successive observed failures.
Let

$$t_0 = 0$$

$$t_i = \inf\{t > t_{i-1} | M(t) \neq M(t_{i-1})\}$$

$$\tau_i = t_i - t_{i-1}; \quad i = 1, 2, \ldots, N$$

On $t_1 > s > 0$, we have

$$\tau_1 \circ \vartheta_s = \inf\{t > 0 | M(t+s) \neq M(0+)\}$$

$$\tau_1 \circ \vartheta_s + s = \inf\{t > 0 | M(t+s) \neq M(0+)\} + s$$

$$= \inf\{t + s > s | M(t+s) \neq M(0+)\} = \tau_1$$

So

$$P_{\langle M(0), A(0+)\rangle}\{\tau_1 > t + s\} = P_{\langle M(0), A(0+)\rangle}\{\tau_1 > t + s, \tau_1 > s\}$$

$$= P_{\langle M(0), A(0+)\rangle}\{\tau_1 \circ \vartheta_s > t, \tau_1 > s\}$$

$$= E_{\langle M(0), A(0+)\rangle}\left[P_{\langle M(0), A(0+)\rangle}\{\tau_1 > t\} I_{\{\tau_1 > s\}}\right]$$

where

$$I_{\{\tau_1 > s\}}(\omega) = \begin{cases} 1 & \text{if } \omega \in \{\tau_1 > s\} \\ 0 & \text{otherwise} \end{cases}$$

In this way

$$P_{\langle M(0), A(0+)\rangle}\{\tau_1 > t + s\} = E_{\langle M(0), A(0+)\rangle}\lfloor P_{\langle M(0), A(s+)\rangle}\{\tau_1 > t\} I_{\{\tau_1 > s\}}\rfloor$$

$$= \sum_{k=1}^{m} P_{\langle M(0), k\rangle}\{\tau_1 > t\} P_{\langle M(0), A(0+)\rangle}\{\tau_1 > s, A(s+) = k\}$$

Suppose

$$P_{\langle M(0), k\rangle}\{\tau_1 > t\} \neq P_{\langle M(0), l\rangle}\{\tau_1 > t\}; \quad \forall k \neq l$$

holds for some $t \in [0, \infty)$, or there exists a pair of actions k and l such that $\theta_k \neq \theta_l$ (refer to Remark (1) of Section 8.2), then τ_1 is not exponentially distributed.

However we can show that τ_1 approaches an exponential distribution for sufficiently large t. Let

$$\hbar_s(M(0)) = \max_{1 \leq k \leq m} P_{\langle M(0), k\rangle}\{\tau_1 > s\}$$

Then we have

$$P_{\langle M(0), A(0+)\rangle}\{\tau_1 > t + s\} \leq \hbar_s(M(0)) P_{\langle M(0), A(0+)\rangle}\{\tau_1 > s\}$$

This implies

$$\hbar_{s+t}(M(0)) \leq \hbar_s(M(0))\hbar_t(M(0))$$

In this way

$$\lim_{t\to\infty}\frac{1}{t}\ln\hbar_t(M(0)) = \inf_{t>0}\left\{\frac{\ln\hbar_t(M(0))}{t}\right\}$$

Therefore

$$\hbar_t(M(0)) \approx \mathrm{e}^{-\lambda_{M(0)}t} \quad \text{for sufficiently large } t$$

where

$$\lambda_{M(0)} = -\inf_{t>0}\left\{\frac{\ln\hbar_t(M(0))}{t}\right\}$$

Similarly, for $\tau_i, \quad i = 2, 3, \ldots, N$, we have

$$\hbar_t(M(t_{i-1})) \approx \mathrm{e}^{-\lambda_{M(t_{i-1})}t} \quad \text{for sufficiently large } t$$

where

$$\lambda_{M(t_{i-1})} = -\inf_{t>0}\left\{\frac{\ln\hbar_t(M(t_{i-1}))}{t}\right\}$$

From this we can roughly say that the exponentiality assumption that times between successive observed failures are exponentially distributed tends to hold as software testing proceeds. This property can be treated as a kind of inner stability of software testing as discussed in Section 6.

8.4.2 Behavior of the times between successive distinct actions.

Let

$$\begin{aligned}
\ell_0 &= 0\\
\ell_i &= \inf\{t > \ell_{i-1} | A(t+) \neq A(\ell_{i-1})\}\\
\varsigma_i &= \ell_i - \ell_{i-1}; \quad i = 1, 2, \ldots, N
\end{aligned}$$

Similar to Section 8.4.1, we can have

$$\begin{aligned}
P_{\langle M(0),A(0+)\rangle}\{\varsigma_1 > t+s\} &= P_{\langle M(0),A(0+)\rangle}\{\varsigma_1\circ\vartheta_s + s > t+s, \varsigma_1 > s\}\\
&= P_{\langle M(0),A(0+)\rangle}\{\varsigma_1\circ\vartheta_s > t, \varsigma_1 > s\}\\
&= E_{\langle M(0),A(0+)\rangle}\left[P_{\langle M(s),A(s+)\rangle}\{\varsigma_1 > s\}I_{\{\varsigma_1>s\}}\right]\\
&= \sum_{j=1}^{N} P_{\langle j,A(0+)\rangle}\{\varsigma_1 > t\}P_{\langle M(0),A(0+)\rangle}\{\varsigma_1 > t+s, M(s) = j\}
\end{aligned}$$

Let

$$\mu_t(A(0+)) = \max_{1\le j\le N} P_{\langle j,A(0+)\rangle}\{\varsigma_1 > t\}$$

Then we obtain

$$\lim_{t\to\infty} \frac{\ln \mu_t(A(0+))}{t} = -\lambda_{A(0+)} < 0$$

$$\mu_t(A(0+)) \approx \mathrm{e}^{-\lambda_{A(0+)}t} \quad \text{for sufficiently large } t$$

where

$$\lambda_{A(0+)} = -\inf_{t>0}\left\{\frac{\ln \mu_t(A(0+))}{t}\right\}$$

That is, ς_1 tends to be exponentially distributed and the software testing process demonstrates a kind of inner stability.

Similarly, for $\varsigma_i, \quad i = 2, 3, \ldots, N$, we have

$$\mu_t(M(\ell_{i-1})) \approx \mathrm{e}^{-\lambda_{M(\ell_{i-1})}t} \quad \text{for sufficiently large } t$$

where

$$\lambda_{M(\ell_{i-1})} = -\inf_{t>0}\left\{\frac{\ln \hbar_t(M(\ell_{i-1}))}{t}\right\}$$

8.4.3 Behavior of the times between successive observed failures or distinct actions.

Let

$$\chi_0 = 0$$

$$\chi_i = \inf\{t > \chi_{i-1} | \langle M(t), A(t)\rangle \neq \langle M(\chi_{i-1}), A(\chi_{i-1}+)\rangle\}$$

$$\eta_i = \chi_i - \chi_{i-1}; \quad i = 1, 2, \ldots, N$$

$$t_0 = 0$$

$$t_i = \inf\{t > t_{i-1} | M(t) \neq M(t_{i-1})\}$$

$$\tau_i = t_i - t_{i-1}; \quad i = 1, 2, \ldots, N$$

$$\ell_0 = 0$$

$$\ell_i = \inf\{t > \ell_{i-1} | A(t+) \neq A(\ell_{i-1})\}$$

$$\varsigma_i = \ell_i - \ell_{i-1}; \quad i = 1, 2, \ldots, N$$

Obviously,

$$\chi_i = \min\{t_i, \ell_i\}; \quad i = 0, 1, \ldots, N$$

We have

$$\begin{aligned} P_{\langle M(0),A(0+)\rangle}\{\chi_1 > t+s\} &= P_{\langle M(0),A(0+)\rangle}\{\chi_1 \circ \vartheta_s + s > t+s, \chi_1 > s\} \\ &= P_{\langle M(0),A(0+)\rangle}\{\chi_1 \circ \vartheta_s > t, \chi_1 > s\} \\ &= E_{\langle M(0),A(0+)\rangle}\left[P_{\langle M(s),A(s+)\rangle}\{\chi_1 > t\} I_{\{\chi_1 > s\}}\right] \\ &= E_{\langle M(0),A(0+)\rangle}\left[P_{\langle M(0),A(0+)\rangle}\{\chi_1 > t\} I_{\{\chi_1 > s\}}\right] \\ &= P_{\langle M(0),A(0+)\rangle}\{\chi_1 > t\} P_{\langle M(0),A(0+)\rangle}\{\chi_1 > s\} \end{aligned}$$

Therefore χ_1 is exponentially distributed with respect to $\{\langle M(t), A(t)\rangle, t \geq 0\}$. There exists a parameter $\lambda_{\langle M(0),A(0+)\rangle} > 0$ such that

$$P_{\langle M(0),A(0+)\rangle}\{\chi_1 > t\} = \mathrm{e}^{-\lambda_{\langle M(0),A(0+)\rangle t}}$$

From the fact that

$$P_{\langle M(0),A(0+)\rangle}\{\tau_1 > t\} > P_{\langle M(0),\, A(0+)\rangle}\{\chi_1 > t\}$$

we arrive at

$$\lambda_{M(0)} < \lambda_{\langle M(0),\, A(0+)\rangle}$$

Similarly, we can obtain

$$\lambda_{A(0+)} < \lambda_{\langle M(0),\, A(0+)\rangle}$$

Similarly, we can claim that $\chi_2, \chi_3, \ldots, \chi_N$ are all exponentially distributed with respect to $\{\langle M(t), A(t)\rangle, t \geq 0\}$.

9. A generalized model of software reliability testing

As pointed out in Section 2.3, there are threats to the validity of the simplifying model presented in Section 2.2. The major threats are that detected defects are often not removed immediately and defects are not equally detectable. These threats can be coped by adopting a distinct set of model assumptions.

9.1 Model assumptions

The assumptions for the generalized model of software reliability testing are as follows.

(1) The input domain or the given test suite, C, of the software under test comprises m classes of test cases, $C_1, C_2, \ldots, C_m$, which may or may not be disjoint; that is, $C = \bigcup_{j=1}^{m} C_j$; $C_1, C_2, \ldots, C_m$ do not change in the course of software testing.
(2) The software under test contains N defects at the beginning of testing.
(3) Each of the N defects is in one of three distinct states at any time: removed or absent from the software under test, undetected by any action, or detected but not removed

from the software under test; symbolically, let

$$Y_i^{(k)} = \begin{cases} 0 & \text{if the } k\text{th defect has been removed from} \\ & \quad \text{the software when the } i\text{th action is applied} \\ 1 & \text{if the } k\text{th defect remains undetected in the first } i \text{ actions} \\ 2 & \text{if the } k\text{th defect is detected by some of the first } i \text{ actions,} \\ & \quad \text{but has not been removed from the software under test} \end{cases};$$

$k = 1,2,\ldots,N; \quad i = 1,2,\ldots$ with

$$Y_0^{(1)} = Y_0^{(2)} = \cdots = Y_0^{(N)} = 1$$

(4) Each action detects at most one defect; a new failure is revealed if an action detects a defect that was not detected by previous actions. Symbolically, suppose the current action the ith action, then let

$$Z_i = \begin{cases} 1 & Y_i^{(k)} > Y_{i-1}^{(k)} \quad \text{for some } k \\ 0 & \text{otherwise} \end{cases}$$

(5) Upon a total of d new failure being revealed, the corresponding d failure-causing defects are removed immediately and instantaneously from the software under test, and no new defects are introduced. Symbolically, suppose the current action the ith action, then $Y_i^{(k)}$ is updated as follows

$$Y_i^{(k)} = \begin{cases} 0 & \text{if } Y_i^{(k)} = 0 \text{ or } 2 \\ 1 & \text{if } Y_i^{(k)} = 1 \end{cases}$$

The defect removals do not affect the probabilities of an action detecting undetected defects.

(6) A next test case is selected and executed after the current action is finished; the sequence $\{A_1, A_2, \ldots, A_i, A_{i+1}, \ldots\}$ forms a Markov chain with

$$\Pr\{A_{i+1} = l | A_i = k\} = p_{kl}$$

(7) During the time interval $[0, t)$ a total of $H(t) + 1$ test cases are selected; the first one is taken at the beginning of software testing and $H(t)$ forms a Poisson process with parameter λ, or

$$\Pr\{H(t) = k\} = \frac{(\lambda t)^k}{k!} e^{\lambda t}; \quad k = 0, 1, 2, \ldots$$

where λ is referred to as the testing intensity; each testing action including the first one takes an exponentially distributed length of time with parameter λ.

(8) The first i actions or test cases reveals M_i failures and the testing process during the time interval $[0, t]$ removes $M(t)$ defects; that is,

$$M_i = \sum_{k=1}^{i} Z_k \quad \text{with } M_0 = 0$$

$$M(t) = \sum_{k=0}^{H(t)} Z_k \quad \text{with } M(0) = 0, \quad Z_0 = 0$$

(9) The probability of a test case detecting a defect is determined as follows,

$$\Pr\{Y_i^{(k)} = 0 | A_i = j, Y_i^{(k)} = 0\} = 1$$
$$\Pr\{Y_i^{(k)} = 0 | A_i = j, Y_i^{(k)} = 1\} = 0$$
$$\Pr\{Y_i^{(k)} = 1 | A_i = j, Y_i^{(k)} = 1\} = 1 - \theta_{jk}$$
$$\Pr\{Y_i^{(k)} = 2 | A_i = j, Y_i^{(k)} = 1\} = \theta_{jk}$$
$$\Pr\{Y_i^{(k)} = 2 | A_i = j, Y_i^{(k)} = 2\} = 1;$$
$$i = 1, 2, \ldots; \quad k = 0, 1, \ldots, N-1$$

(10) $\{M_1, M_2, \ldots\}$ and $\{A_1, A_2, \ldots\}$ are conditionally independent of each other as follows,

$$\Pr\{M_1, A_2 | A_1\} = \Pr\{M_1 | A_1\} \Pr\{A_2 | A_1\}$$

and for $i > 1$,

$$\Pr\{M_i, A_{i+1} | M_{i-1}, \ldots, M_1, A_i, A_{i-1}, \ldots, A_1\}$$
$$= \Pr\{M_i | M_{i-1}, \ldots, M_1, A_i, A_{i-1}, \ldots, A_1\} \Pr\{A_{i+1} | M_{i-1}, \ldots, M_1, A_i, A_{i-1}, \ldots, A_1\}$$

$$\Pr\{M_i | M_{i-1}, \ldots, M_1, A_i, A_{i-1}, \ldots, A_1\} = \Pr\{M_i | M_{i-1}, A_i\}$$
$$\Pr\{A_{i+1} | M_{i-1}, \ldots, M_1, A_i, A_{i-1}, \ldots, A_1\} = \Pr\{A_{i+1} | A_i\}$$

(11) The process $\{M_i; \ i = 0, 1, 2, \ldots\}$ is independent of the Poisson process $\{H(t), t \geq 0\}$; more accurately, it holds

$$\Pr\{M_0 = 0, M_1 = k_1, \ldots, M_i = k_i | H(t) = i\} = \Pr\{M_0 = 0, M_1 = k_1, \ldots, M_i = k_i\}$$

(12) The process $\{A_i; \ i = 0, 1, 2, \ldots\}$ is independent of the Poisson process $\{H(t), t \geq 0\}$; more accurately, it holds

$$\Pr\{A_1 = j_1, A_2 = j_2, \ldots, A_i = j_i | H(t) = i\} = \Pr\{A_1 = j_1, A_2 = j_2, \ldots, A_i = j_i\}$$

(13) The first testing action is selected according to the probability distribution $\{p_1, p_2, \ldots, p_m\}$, that is, $\Pr\{A_1 = j\} = p_j; \quad j = 1, 2, \ldots, m$.

(14) The software testing process terminates till all the N defects are removed.

Remarks.

(1) A major distinction of the present model from the simplifying model is the introduction of the intermediate variable $Y_i^{(k)}$. This variable identifies the state of a defect and relates software defect detection process to software failure observation process. It makes the behavior of the behavior of $\{M_i; i = 0, 1, 2, \ldots\}$ or $\{M(t); t \geq 0\}$ much more complicated than the counterpart of the simplifying model.

(2) Assumption (9) indicates that defects are not necessarily equally detectable. It characterizes the behavior of $Y_i^{(k)}$ rather than Z_i directly.

(3) Assumption (4) implies that there are only N new failures in total. Each defect corresponds to one new failure, although a defect may be detected several times since detected defects may or may not be removed immediately.

(4) The generalized model reduces to the simplifying model if $d = 1$ and $\theta_{jk} \equiv \theta_j; \quad k = 1, 2, \ldots, N; \quad j = 1, 2, \ldots, m.$

(5) More general models can be considered. For example, defect removals may be imperfect; new defects may be introduced while detected defects are removed. The time of executing an test case may not necessarily be exponentially distributed, it may eventually be deterministic. The simplifying and generalized models set an example to show how to model software reliability testing process more rigorously and realistically.

9.2 Simulation model

Due to the introduction of the intermediate variable $Y_i^{(k)}$, it looks mathematically involved to analytically investigate the behavior of $\{M_i; i = 0, 1, 2, \ldots\}$ or $\{M(t); t \geq 0\}$ and the corresponding times between successive distinctions. In order to get a rudimentary judgment of the generalized model presented in last subsection, we do some of simulation. To this end, we need to present a simulation model. The key obstacle to simulate the generalized model is how to represent the input domain and various distinct defect detection rates.

Let $C = \{1, 2, \ldots, K\}$; that is, C comprises K positive integers with the positive integer l representing the lth test case. We divide C into $N + 1$ disjoint parts $D_0, D_1, \ldots, D_N$, where D_0 comprises all test cases that are not capable of detecting any of the N defects, and D_k comprises all the test cases that are capable of detecting the ith defects, $k = 1, 2, \ldots, N$. Since $D_0, D_1, \ldots, D_N$ are disjoint, we implicitly assume that no single test case is capable of detecting two distinct defects for the purpose of simulation. Without loss of generality, we assume $D_k = [n_k, n_k + 1, \ldots, l_k]$, where $n_0 = 1, n_k, l_k \in D, \quad n_k \leq l_k, \quad n_k \leq n_{k+1}$, $i = 0, 1, 2, \ldots, N$.

We further partition D_0 into m disjoint parts $D_{01}, D_{02}, \ldots, D_{0m}$. Let

$$C_j = D_{0j} \cup \left(\bigcup_{k=1}^{N} D_k \right)$$

Then

$$C = C_1 \cup C_2 \cup \cdots \cup C_m = \left(\bigcup_{k=0}^{N} D_k \right) = \left(\bigcup_{j=1}^{m} D_{0j} \right) \cup \left(\bigcup_{k=1}^{N} D_k \right)$$

Suppose C_j represents class or distinct action j. Then a test case can be selected or generated in two steps:

(1) A class is selected from $\{C_1, C_2, \ldots, C_m\}$ in accordance with the given Markov chain such that

$$\Pr\{A_{i+1} = l | A_i = k\} = p_{kl}$$

The first action is selected in accordance with the probability distribution $\{p_1, p_2, \ldots, p_m\}$; Suppose C_j is selected;

(2) A test case or a positive integer is picked up from C_j in accordance with a uniform probability distribution; the corresponding probability of D_k being selected or the kth defects being detected (if it has not been removed) is

$$\theta_{jk} = \frac{|D_k|}{|D_{0j}| + |D_1| + |D_2| + \cdots + |D_N|}$$

where $|D|$ denotes the cardinality of set D. By assigning different sizes to various sets, θ_{jk} may vary with j and k.

Remarks.

(1) Due to the space limitation, a complete description of the required simulation model is avoided here. However readers should be able to understand how the required simulation model can be generated from the model assumptions presented in Section 9.1.
(2) The assumption that each action can only detect at most one defect is not essential. We can partition C such that $D_1, D_2, \ldots, D_N$ may overlap for the purpose of simulation.
(3) In the simulation model the various partitions keep invariant during testing. This makes cardinalities of various sets including $D_{01}, D_{02}, \ldots, D_{0m}$ and $D_0, D_1, \ldots, D_N$ remain constant and thus the defect detection rates (various θ_{jk}'s) do not change for un-removed defects.
(4) The simulation model presented here can be treated as an *integer model*, where each defect is represented as a set of positive integers. This is different from conventional *urn-ball model* of software testing, where a defect is treated as a white or black ball in an urn (Friedman and Voas 1995). Note that by adopting the integer model, the resulting defect detection rates must be a rational number. In order to consider the possibility of the defect detection rates taking irrational numbers as values, the integer model can be extended to a *real line model*, where each defect is represented as a segment in the real line.

9.3 Simulation examples

In order to get a rough understanding of the behavior of the generalized model, we carry out simulation as in Example 7.1. The parameter settings are chosen rather arbitrarily. There are no obvious rules to follow. The simulation examples are aimed to shed some light on possible behavior of the generalized model. An extensive simulation study should be done in the future.

Example 9.1. The parameter setting of the simulation is as follows.

$$N = 30, \quad m = 3, \quad d = 4, \quad \lambda = 5$$

Initial probability distribution for A_1:

0.53872680735715	0.17595323667631	0.28531995596654

Transition probabilities:

0.05118612109273	0.27054213798779	0.67827174091948
0.30101313176612	0.08926453263789	0.60972233559599
0.34790469768825	0.15275596323910	0.49933933907265

$$\{\theta_{jk}; j = 1, 2, 3; k = 1, 2, \ldots, 30\}$$

0.0033870	0.0055249	0.0010013
0.0084674	0.0138122	0.0025031
0.0093141	0.0151934	0.0027534
0.0067739	0.0110497	0.0020025
0.0008467	0.0013812	0.0002503
0.0169348	0.0276243	0.0050063
0.0033870	0.0055249	0.0010013
0.0194750	0.0317680	0.0057572
0.0025402	0.0041436	0.0007509
0.0101609	0.0165746	0.0030038
0.0033870	0.0055249	0.0010013
0.0008467	0.0013812	0.0002503
0.0008467	0.0013812	0.0002503
0.0008467	0.0013812	0.0002503
0.0059272	0.0096685	0.0017522
0.0211685	0.0345304	0.0062578
0.0025402	0.0041436	0.0007509
0.0313294	0.0511050	0.0092616
0.0059272	0.0096685	0.0017522
0.0025402	0.0041436	0.0007509
0.0016935	0.0027624	0.0005006
0.0177815	0.0290055	0.0052566
0.0050804	0.0082873	0.0015019
0.0033870	0.0055249	0.0010013
0.0016935	0.0027624	0.0005006
0.0313294	0.0511050	0.0092616
0.0008467	0.0013812	0.0002503
0.0152413	0.0248619	0.0045056
0.0033870	0.0055249	0.0010013
0.0169348	0.0276243	0.0050063

Figures 4 and 5 show the behavior of $EM(t)$ and $-\ln(1-(EM(t)/N))$ up to $t = 1000$, respectively. In comparison with Example 7.1, $-\ln(1-(EM(t)/N))$ is no longer a linear function of t and thus $EM(t)$ does not fit an exponential law. However we can see that $-\ln(1-EM(t)/N)$ is close to a piece-wise linear function comprising two segments. This implies that after an initial testing stage, $EM(t)$ tends to fit an exponential law. The major difference between the behavior of $EM(t)$ presented in Example 7.1 and that demonstrated in figures 4 and 5 lies in the initial testing stage. This suggests that although the simplifying model is different from the generalized model and deviates from the software testing engineering

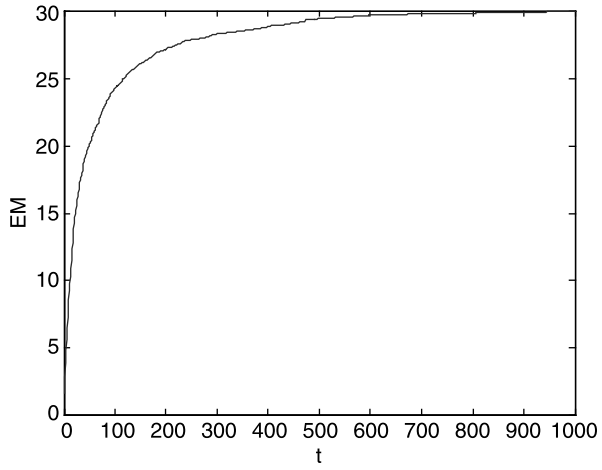

Figure 4. Simulation results of $EM(t)$ for example 9.1.

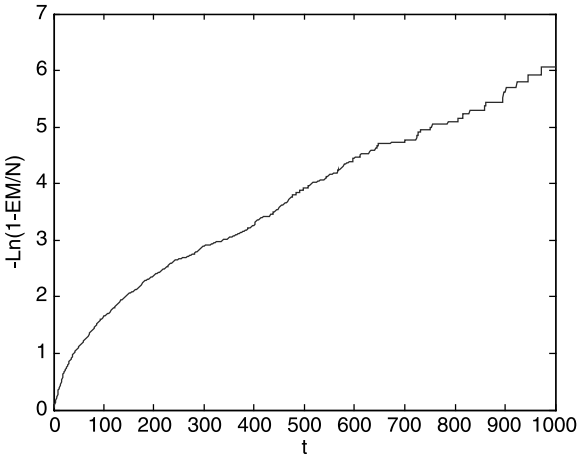

Figure 5. Simulation results of $-\ln(1-(EM(t)/N))$ for example 9.1.

practice, it does capture some essential properties of the software testing process in the late testing stage and thus makes sense.

Example 9.2. The parameter setting of the simulation is as follows.

$$N = 50, \quad m = 3, \quad d = 4, \quad \lambda = 5$$

Initial probability distribution for A_1:

0.05163574314287	0.79846106265629	0.14990319420084

Transition probabilities:

0.40158149479031	0.22750850707389	0.37090999813580
0.73468417568816	0.22205363584216	0.04326218846968
0.46387293025100	0.16745324733513	0.36867382241387

$$\{\theta_{jk}; j = 1, 2, 3; k = 1, 2, \ldots, 30\}$$

0.0016071	0.0032841	0.0018240
0.0008035	0.0016420	0.0009120
0.0008035	0.0016420	0.0009120
0.0008035	0.0016420	0.0009120
0.0016071	0.0032841	0.0018240
0.0044194	0.0090312	0.0050160
0.0032141	0.0065681	0.0036480
0.0004018	0.0008210	0.0004560
0.0020088	0.0041051	0.0022800
0.0032141	0.0065681	0.0036480
0.0004018	0.0008210	0.0004560
0.0020088	0.0041051	0.0022800
0.0004018	0.0008210	0.0004560
0.0016071	0.0032841	0.0018240
0.0012053	0.0024631	0.0013680

0.0012053	0.0024631	0.0013680
0.0060265	0.0123153	0.0068399
0.0008035	0.0016420	0.0009120
0.0012053	0.0024631	0.0013680
0.0012053	0.0024631	0.0013680
0.0036159	0.0073892	0.0041040
0.0016071	0.0032841	0.0018240
0.0004018	0.0008210	0.0004560
0.0004018	0.0008210	0.0004560
0.0004018	0.0008210	0.0004560
0.0012053	0.0024631	0.0013680
0.0016071	0.0032841	0.0018240
0.0044194	0.0090312	0.0050160
0.0020088	0.0041051	0.0022800
0.0036159	0.0073892	0.0041040
0.0012053	0.0024631	0.0013680
0.0144636	0.0295567	0.0164159
0.0004018	0.0008210	0.0004560
0.0028124	0.0057471	0.0031920
0.0012053	0.0024631	0.0013680
0.0008035	0.0016420	0.0009120
0.0048212	0.0098522	0.0054720
0.0036159	0.0073892	0.0041040
0.0024106	0.0049261	0.0027360
0.0008035	0.0016420	0.0009120
0.0008035	0.0016420	0.0009120
0.0008035	0.0016420	0.0009120
0.0144636	0.0295567	0.0164159
0.0004018	0.0008210	0.0004560
0.0004018	0.0008210	0.0004560
0.0044194	0.0090312	0.0050160
0.0028124	0.0057471	0.0031920
0.0016071	0.0032841	0.0018240
0.0016071	0.0032841	0.0018240
0.0064283	0.0131363	0.0072959

Figures 6 and 7 show the behavior of $EM(t)$ and $-\ln(1-(EM(t)/N))$ up to $t = 1500$, respectively. They coincide with figures 4 and 5 in that after an initial testing stage, $EM(t)$ tends to fit an exponential law, although the behavior of $-\ln(1-(EM(t)/N))$ in the late testing stage demonstrate minor fluctuations. □

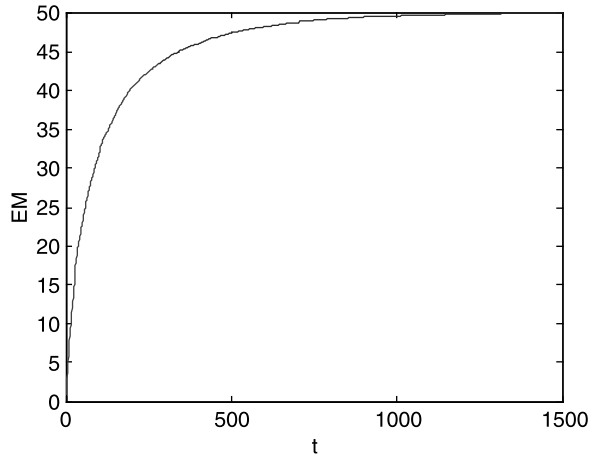

Figure 6. Simulation results of $EM(t)$ for example 9.2.

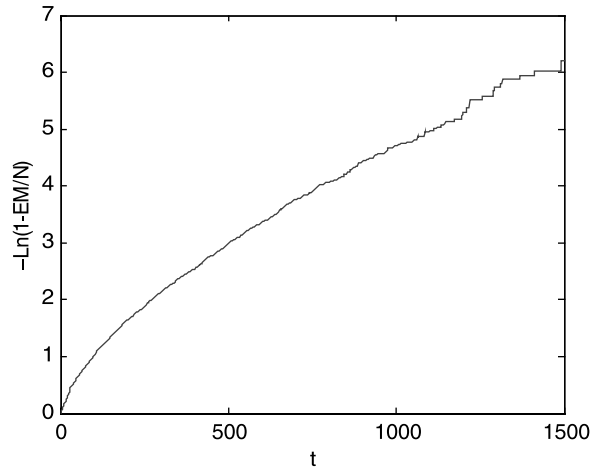

Figure 7. 9.4 Simulation results of $-\ln(1-(EM(t)/N))$ for example 9.2.

Example 9.3. The parameter setting of the simulation is as follows.

$$N = 100, \quad m = 3, \quad d = 4, \quad \lambda = 5$$

Initial probability distribution for A_1:

0.56454467892858	0.13168985506120	0.30376546601022

Transition probabilities:

0.75197686848788	0.11786770437175	0.13015542714037
0.16835521961020	0.45422441013116	0.37742037025864
0.57984116281376	0.16403984627265	0.25611899091359

$$\{\theta_{jk}; j = 1, 2, 3; k = 1, 2, \ldots, 30\}$$

0.0033113	0.0005755	0.0011969
0.0016556	0.0002878	0.0005984
0.0016556	0.0002878	0.0005984
0.0008278	0.0001439	0.0002992
0.0024834	0.0004317	0.0008977
0.0041391	0.0007194	0.0014961
0.0024834	0.0004317	0.0008977
0.0016556	0.0002878	0.0005984
0.0016556	0.0002878	0.0005984
0.0024834	0.0004317	0.0008977
0.0016556	0.0002878	0.0005984
0.0041391	0.0007194	0.0014961
0.0016556	0.0002878	0.0005984
0.0016556	0.0002878	0.0005984
0.0033113	0.0005755	0.0011969
0.0057947	0.0010072	0.0020946
0.0016556	0.0002878	0.0005984
0.0024834	0.0004317	0.0008977
0.0008278	0.0001439	0.0002992
0.0008278	0.0001439	0.0002992

0.0057947	0.0010072	0.0020946
0.0016556	0.0002878	0.0005984
0.0107616	0.0018705	0.0038899
0.0016556	0.0002878	0.0005984
0.0066225	0.0011511	0.0023938
0.0008278	0.0001439	0.0002992
0.0066225	0.0011511	0.0023938
0.0016556	0.0002878	0.0005984
0.0024834	0.0004317	0.0008977
0.0041391	0.0007194	0.0014961
0.0008278	0.0001439	0.0002992
0.0049669	0.0008633	0.0017953
0.0099338	0.0017266	0.0035907
0.0008278	0.0001439	0.0002992
0.0033113	0.0005755	0.0011969
0.0024834	0.0004317	0.0008977
0.0049669	0.0008633	0.0017953
0.0008278	0.0001439	0.0002992
0.0033113	0.0005755	0.0011969
0.0124172	0.0021583	0.0044883
0.0024834	0.0004317	0.0008977
0.0008278	0.0001439	0.0002992
0.0016556	0.0002878	0.0005984
0.0016556	0.0002878	0.0005984
0.0008278	0.0001439	0.0002992
0.0024834	0.0004317	0.0008977
0.0008278	0.0001439	0.0002992
0.0024834	0.0004317	0.0008977
0.0008278	0.0001439	0.0002992
0.0049669	0.0008633	0.0017953
0.0132450	0.0023022	0.0047876
0.0016556	0.0002878	0.0005984
0.0008278	0.0001439	0.0002992
0.0033113	0.0005755	0.0011969
0.0033113	0.0005755	0.0011969
0.0033113	0.0005755	0.0011969
0.0041391	0.0007194	0.0014961
0.0008278	0.0001439	0.0002992
0.0041391	0.0007194	0.0014961
0.0008278	0.0001439	0.0002992
0.0033113	0.0005755	0.0011969
0.0016556	0.0002878	0.0005984
0.0049669	0.0008633	0.0017953
0.0099338	0.0017266	0.0035907
0.0140728	0.0024460	0.0050868
0.0107616	0.0018705	0.0038899
0.0057947	0.0010072	0.0020946
0.0016556	0.0002878	0.0005984
0.0016556	0.0002878	0.0005984
0.0057947	0.0010072	0.0020946
0.0049669	0.0008633	0.0017953
0.0033113	0.0005755	0.0011969
0.0008278	0.0001439	0.0002992
0.0016556	0.0002878	0.0005984
0.0008278	0.0001439	0.0002992
0.0165563	0.0028777	0.0059844
0.0091060	0.0015827	0.0032914
0.0024834	0.0004317	0.0008977
0.0016556	0.0002878	0.0005984
0.0057947	0.0010072	0.0020946
0.0016556	0.0002878	0.0005984
0.0008278	0.0001439	0.0002992
0.0016556	0.0002878	0.0005984
0.0033113	0.0005755	0.0011969
0.0016556	0.0002878	0.0005984
0.0149007	0.0025899	0.0053860

0.0024834	0.0004317	0.0008977
0.0016556	0.0002878	0.0005984
0.0273179	0.0047482	0.0098743
0.0033113	0.0005755	0.0011969
0.0024834	0.0004317	0.0008977
0.0008278	0.0001439	0.0002992
0.0149007	0.0025899	0.0053860
0.0099338	0.0017266	0.0035907
0.0016556	0.0002878	0.0005984
0.0024834	0.0004317	0.0008977
0.0008278	0.0001439	0.0002992
0.0057947	0.0010072	0.0020946
0.0165563	0.0028777	0.0059844
0.0057947	0.0010072	0.0020946

Figures 8 and 9 show the behavior of $EM(t)$ and $-\ln(1 - EM(t)/N)$ up to $t = 1500$, respectively. As in Examples 9.1 and 9.2, $EM(t)$ tends to fit an exponential law after an initial testing stage. □

Overall, the major conclusion that can be drawn from the above simulation examples is that the simplifying model presented in Section 2 make sense particularly after a short initial

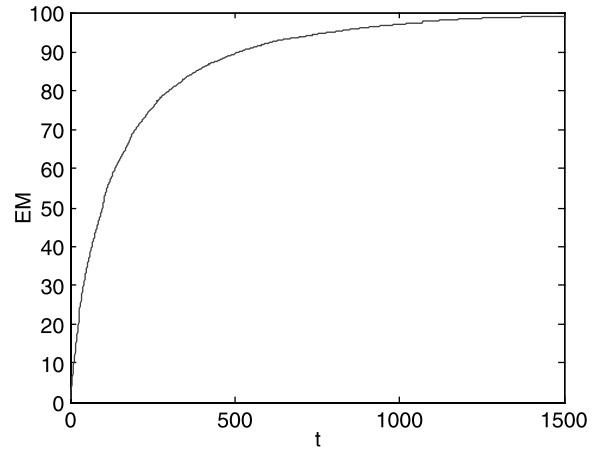

Figure 8. Simulation results of $EM(t)$ for example 9.3.

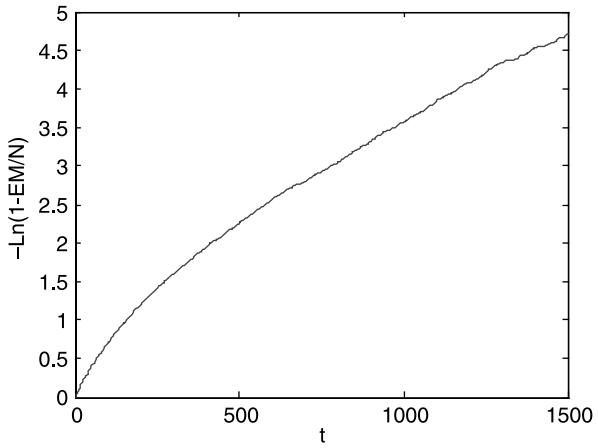

Figure 9. Simulation results of $-\ln(1 - (EM(t)/N))$ for example 9.3.

testing stage and the generalized model describes more accurately the software reliability testing process for the short initial testing stage. For various software reliability testing processes, $EM(t)$ always tends to fit an exponential law after a short initial testing stage.

10. General discussion

Up to this point we can have a general discussion about the modeling framework. What can be drawn from various specific theoretical and simulation results presented in the preceding sections? How important are these results in general? How can they be improved in the future? We should clarify what we have achieved and what future work should be conducted.

10.1 Characteristics of the modeling framework

The general requirement of a desirable modeling framework is that the modeling framework should be realistic, rigorous and quantitative. The modeling framework comprising the simplifying model and the generalized model satisfies this general requirement to a large extent. More specifically, the modeling framework is featured with the following characteristics:

(1) Software test case selections and defect removal mechanisms are explicitly formulated.
(2) Both continuous-time domain and discrete-time domain are considered; the relation between these two domains is well described.
(3) The unrealistic assumption that software test profile coincides with software operational profile is avoided.
(4) Two major reliability measures, the cumulative number of observed failures and the times between successive distinctions, are quantified in a mathematically rigorous manner.

The only doubt that one may cast is that how much the modeling framework is close to the software testing engineering practice. Example 7.1 and the simulation example presented in Section 9.3 should give positive judgment of the rationale of the modeling framework. Of course, the validity of the modeling framework must be tested in the future case studies.

10.2 General conclusions

A few general conclusions can be drawn from the results presented in the preceding sections.

(1) The software reliability testing process can really be modeled in a practically realistic, mathematically rigorous, and quantitatively precise manner, although this is a challenging task.
(2) For the simplifying model, $\{\langle M_i, A_i\rangle; i = 1, 2, \dots\}$ is a Markov chain and $\{\langle M(t), A(t)\rangle; t \geq 0\}$ is a Markov process. However, $\{M_i, i = 1, 2, \dots\}$ is not a Markov chain in general and thus $\{M(t); t \geq 0\}$ is not a Markov process in general either.
(3) The NHPP assumption is theoretically false, although it has widely been adopted in software reliability modeling. However $EM(t)$ fits an exponential law for specific cases. It may deviate from an exponential law in general but tend to approach an exponential law asymptotically as software testing proceeds.
(4) The exponentiality assumption does not hold in general. However it holds for specific cases and tends to hold asymptotically as software testing proceeds in general.

(5) The independence assumption does not hold in general, although it does not hold for some specific cases.
(6) The notion of stability should be introduced to characterize the process of software testing and may be of vast potential in theory as well as in practice.

10.3 Major contributions of this paper

The major contributions of this paper can be summarized as follows.

(1) A realistic, rigorous, and quantitative modeling framework is introduced for the software reliability testing process.
(2) Several theoretical conclusions are drawn for software reliability testing.

However one may doubt the values of these contributions. Is a modeling framework necessary or superfluous? How are theoretical conclusions useful? To alleviate these doubts, we just note that software testing is probably the least understood part of software development process and cost about half of software development resources. Any improved understanding of the software reliability testing process has potential for cost reduction of the software reliability testing and for higher delivered software reliability. A realistic, rigorous, and quantitative modeling framework may guide software process management to make the process more manageable, repeatable and predictable. For example, the formulation of the behavior of $\{M(t); t \geq 0\}$ may help software manager to forecast how much time is required to achieve a desirable reliability goal. It may also be useful for assessing the effectiveness of the test profile determined by $\{p_1, p_2, \ldots, p_m\}$ and $[p_{kl}]$.

From the theoretical viewpoint, theoretical observations improve our understanding of the software reliability testing process and judge or refute some of existing arguments. For example, the observation that the NHPP assumption is theoretically false explains why the widely adopted NHPP framework of software reliability modeling is not appropriate. The observation that $EM(t)$ tends to fit an exponential law reveals that some of related works in the NHPP framework of software reliability modeling still makes sense since the NHPP framework assumes that $EM(t)$ follows an exponential law. More importantly, various theoretical observations may be useful for establishing a systematic theory of software reliability which is yet to exist. We note that conventional system reliability mathematics explores various lifetime distributions and the relationships between component reliabilities and system reliability (Barlow and Proschan 1965, 1975), where the notion of component state or system state plays a fundamental role as demonstrated in the notion of coherent structure. However the notion of software component is not well defined and it is over-simplifying to investigate the relationships between software component reliabilities and system reliabilities from a single perspective of software state. Software system reliability is not only a function of software component reliabilities, but also a function of software execution process since different execution process invokes different set of software components. On the other hand, as a result of the fact that software correctness can hardly be proved mathematically, the software development process including the software reliability testing process have to be taken into account for software reliability assessment and forecast. Software system reliability is a function of the software reliability testing process too.

The potential of introducing a notion of stability for software testing offers us a chance to have systematic examination of various related phenomenon in software testing. It is widely recognized that the software testing process tends to saturate and it is important to identify the saturation points and switch thereafter from one testing scenario or technique to another for the purpose of software testing process improvement (Horgan and Mathur 1996, Menzies *et al.* 2002). Software testing saturation can be treated as an example of software testing stability and thus a notion of software testing stability provides a theoretical framework for investigating the related phenomenon in depth.

10.4 Future work

A lot of works should be carried out in the future for the modeling framework, and some of them are as follows.

(1) Since the modeling framework presented in this paper is mainly mathematical and theoretical, a major challenge is to put it into a software testing practice setting to test its validity and to help software testing process improvement. Case studies must be carried out to justify the modeling framework.
(2) Further theoretical investigation of the simplifying model should be conducted. What are the necessary conditions for $\{M(t); t \geq 0\}$ to be a Markov process? What is the behavior of $EM(t)$ in general?
(3) The theoretical aspect of the generalized model has not been approached in this paper and this must be coped in the future.
(4) More realistic models should be developed for the software reliability testing process. For example, the time consumed by a testing action may not be exponentially distributed. No single model can be universally valid.
(5) The discussion concerning the notion of software testing stability presented in this paper is only rudimentary; this topic should thoroughly investigated in the future.
(6) Eigenvalue like measures should be introduced to characterize the capability of a testing strategy. This should become a vigorous topic in the modeling framework of software reliability testing.
(7) Decision making schemes should be included in the modeling framework to guide test case generation and selection on-line. For example, how to adjust the testing action transition probabilities $\lfloor p_{ij} \rfloor$ on-line? A testing objective should be explicitly formulated.

11. Concluding remarks

Up to this point we have presented a modeling framework for software reliability testing, comprising the simplifying model and the generalized model. In both models software testing action selection process and the defect removal mechanism are explicitly described. Both the discrete-time domain and the continuous-time domain are involved. The generalized model is more accurate than the simplifying model since the former avoids the assumption that defects are equally detectable and the assumption that defects are removed upon being detected. However simulation examples show that the simplifying model really captures some of essential features of the software testing process after a short initial testing stage.

The modeling framework is practically realistic, mathematically rigorous, and quantitatively precise. It leads to the rigorous examinations of several common assumptions adopted in software reliability modeling, including the independence assumption, the exponentiality assumption, and the NHPP assumption. These assumptions are theoretically false in general. The importance of this work can be justified from several perspectives. First, the software testing process and its relation to software reliability is far beyond being well understood and better understanding of them will lead to software testing process improvement. Second, the paper demonstrates that the important relationship between software testing and software reliability can really be formalized and quantified. Third, theoretical insights can be obtained to justify or refute various existing arguments or assumptions adopted in software reliability testing. Finally, this paper sets a good starting point to have better understanding of software reliability testing.

Acknowledgements

The authors are grateful to anonymous reviewers for their helpful comments which lead to improvements in the readability of this paper.

References

A. Avritzer and E.J. Weyuker, "The automatic generation of load test suites and the assessment of the resulting software", *IEEE Trans. Softw. Eng.*, 21(9), pp. 705–716, 1995.
R.E. Barlow and F. Proschan, *Mathematical Theory of Reliability*, New York: John Wiley & Sons, 1965.
R.E. Barlow and F. Proschan, *Statistical Theory of Reliability and Life Testing: Probability Models*, Holt: Rinbart and Winston, 1975.
B. Beizer, *Software Testing Techniques*, 2nd ed., New York: Van Nostrand Reinhold, 1990.
M. Beyer, W. Dulz and F. Zhen, "Automated TTCN-3 test case generation by means of UML sequence diagrams and Markov chains", *Proceedings of the 12th Asian Test Symposium*, IEEE Computer Society Press, 2003.
R.V. Binder, *Testing Object-Oriented Systems: Models, Patterns, and Tools*, Reading, MA: Addison-Wesley, 2000.
K.Y. Cai, *Software Defect and Operational Profile Modeling*, Dordrecht: Kluwer Academic Publishers, 1998.
K.Y. Cai, "Towards a conceptual framework of software run reliability modeling", *Inf. Sci.*, 126, pp. 137–163, 2000.
K.Y. Cai, "Optimal software testing and adaptive software testing in the context of software cybernetics", *Inf. Softw. Technol.*, 44, pp. 841–855, 2002.
K.Y. Cai, D.B. Hu, C.G. Bai, H. Hu and T. Jing, "Does software reliability growth behavior follow a non-homogeneous poisson process", Submitted for publication, 2004.
K.Y. Cai, Y.C. Li and K. Liu, "Optimal and adaptive testing for software reliability assessment", *Inf. Softw. Technol.*, 46, pp. 989–1000, 2004.
K.Y. Cai, Y.C. Li and W.Y. Ning, "Optimal software testing in the setting of controlled Markov chains", *Eur. J. Oper. Res.*, 162(2), pp. 552–579, 2005.
M.F. Chen, *Eigenvalues, Inequalities and Ergodic Theory*, Beijing: Beijing Normal University, 2003.
M.H. Chen, A.P. Mathur and V.J. Rego, "Effect of testing techniques on software reliability estimates obtained using a time-domain model", *IEEE Trans. Rel.*, 44(1), pp. 97–103, 1995.
S. Chen and S. Mills, "A binary Markov process model for random testing", *IEEE Trans. Softw. Eng.*, 22(3), pp. 218–223, 1996.
K. Doerner and W.J. Gutjahr, "Representation and optimization of software usage models with non-markovian state transitions", *Inf. Softw. Technol.*, 42, pp. 873–887, 2000.
K. Doerner, W.J. Gutjahr, "Extracting test sequences from a Markov software usage model by ACO", in *Genetic and Evolutionary Computation—GECCO 2003*, E. Cantu-Paz, *et al.* Eds, LNCS 2724, Springer, 2003, pp. 2465–2476.
S. Dolev, *Self-Stabilization*, The MIT Press, 2000.
W. Feller, *An Introduction to Probability Theory and Its Applications*, 2nd ed., Vol. II, Wiley, 1971, p. 345.
P.G. Frankl, R.G. Hamlet, B. Littlewwod and L. Strigini, "Evaluating testing methods by delivered reliability", *IEEE Trans. Softw. Eng.*, 24(8), pp. 586–601, 1998.
M.A. Friedman and J.M. Voas, *Software Assessment: Reliability, Safety, Testability*, John Wiley & Sons, 1995.

W.J. Gutjahr, "Importance sampling of test cases in markovian software usage models", *Probability Eng. Inf. Sci.*, 11, pp. 19–26, 1997.
W.J. Gutjahr, "Software dependability evaluation based on Markov usage models", *Perform. Eval.*, 40, pp. 199–222, 2000.
J.R. Horgan and A.P. Mathur, "Software testing and reliability", in *Handbook of Software Reliability Engineering*, M.R. Lyu, Ed., McGraw-Hill, 1996, pp. 531–565.
H.K. Khalil, *Nonlinear Systems*, 3rd ed., Prentice Hall, 2001.
H.J. Kushner, *Stochastic Stability and Control*, Academic Press, 1967.
M.R. Lyu, M.R. Lyu, Ed., *Handbook of Software Reliability Engineering*, McGraw-Hill, 1996.
Y.K. Malaiya and R. Karcich, "The relationship between test coverage and reliability", *Proc. 5th Int. Symp. Softw. Reliability Eng.*, pp. 186–195, 1994.
T. Menzies, D. Owen and B. Cukic, "Saturation effects in testing of formal models", *13th Proc. Int. Symp. Softw. Reliability Eng.*, pp. 15–26, 2002.
J.D. Musa, "Software-reliability-engineered testing", *Computer*, 29(11), pp. 61–68, 1996.
J.H. Poore, G.H. Walton and J.A. Whittaker, "A constraint-basd approach to the representation of software usage models", *Inf. Softw. Technol.*, 42, pp. 825–833, 2000.
R.S. Pressman, *Software Engineering, a Practitioner's Approach*, 5th ed., McGraw-Hill, 2000.
D. Revuz and M. Yor, *Continuous Martingales and Brownian Motion* (A Series of Comprehensive Studies in Mathematics, 293), Springer-Verlag, 1991.
M. Sahinoglu, J. A. von Mayrhauser, A. Hajjar, T. Chen and Ch. Anderson, "On the efficiency of a compound poisson stopping rule for mixed strategy testing", *Proc. IEEE Aerosp. Appl. Conf.*, 1999.
K. Sayre and J.H. Poore, "Partition testing with usage models", *Inf. Softw. Technol.*, 42, pp. 845–850, 2000a.
K. Sayre and J.H. Poore, "Stopping criteria for statistical testing", *Inf. Softw. Technol.*, 42, pp. 851–857, 2000b.
K. Sayre and J. Poore, "A reliability estimator for model based software testing", *Proc. 13th Int. Symp. Softw. Reliability Eng.*, 2002.
G.S. Semmel and D.G. Linton, "Determining optimal testing times for Markov chain usage models", *Proc. IEEE Southeastcon*, 1998.
M.G. Thomason and J.A. Whittaker, "Rare failure state in a Markov chain model for software reliability", *Proc. 10th Int. Symp. Softw. Reliability Eng.*, pp. 12–19, 1999.
G.H. Walton, "Generating transition probabilities for Markov chain usage models", Ph.D. Dissertation, Department of Computer Science, University Tennessee, Knoxville, USA, 1995.
G.H. Walton and J.H. Poore, "Measuring complexity and coverage of software specifications", *Inf. Softw. Technol.*, 42, pp. 859–872, 2000.
J.A. Whittaker, "What is software testing? And why is it so hard?", *IEEE Softw.*, January/February pp. 70–79, 2000.
J.A. Whittaker and J.H. Poore, "Markov analysis of software specifications", *ACM Trans. Softw. Eng. Methodol.*, 2, pp. 93–106, 1993.
J.A. Whittaker and M.G. Thomason, "A Markov chain model for statistical software testing", *IEEE Trans. Softw. Eng.*, 20(10), pp. 812–824, 1994.
J.A. Whittaker, K. Rekab and M.G. Thomason, "A Markov chain model for predicting the reliability of multi-build software", *Inf. Softw. Technol.*, 42, pp. 889–894, 2000.
M. Xie, *Software Reliability Modeling*, World Scientific, 1991.
H. Zhu, P.A. Hall and J.H.R. May, "Test Coverage and Adequacy", *ACM Comput. Surv.*, 29(4), pp. 366–427, 1997.

Appendix

Proof of Proposition 3.3. From the Markovian property of $\{\langle M_i, A_{i+1}\rangle; i = 0, 1, \dots\}$, we have

$$
\begin{aligned}
&\Pr\{M_k = n, A_{k+1} = j; M_{k-1} = n, A_k = i\} \\
&\quad = \Pr\{M_{k+1} = n, M_k = n, A_{k+1} = j; M_{k-1} = n, A_k = i\} \\
&\qquad + \Pr\{M_{k+1} = n+1, M_k = n, A_{k+1} = j; M_{k-1} = n, A_k = i\} \\
&\quad = \Pr\{M_{k+1} = n, A_{k+1} = j | M_{k-1} = n, M_k = n, A_k = i\} \Pr\{M_{k-1} = n, M_k = n, A_k = i\} \\
&\qquad + \Pr\{M_{k+1} = n+1, A_{k+1} = j | M_{k-1} = n, M_k = n, A_k = i\} \Pr\{M_{k-1} = n, M_k = n, A_k = i\} \\
&\quad = \Pr\{M_{k+1} = n, A_{k+1} = j | M_k = n, A_k = i\} \Pr\{M_{k-1} = n, M_k = n, A_k = i\} \\
&\qquad + \Pr\{M_{k+1} = n+1, A_{k+1} = j | M_k = n, A_k = i\} \Pr\{M_{k-1} = n, M_k = n, A_k = i\} \\
&\quad = \{[1 - (N-n)\theta_i] p_{ij} + (N-n)\theta_i p_{ij}\} \Pr\{M_{k-1} = n, M_k = n, A_k = i\} \\
&\quad = p_{ij} \Pr\{M_{k-1} = n, M_k = n, A_k = i\}
\end{aligned}
$$

Note that

$$\Pr\{M_k = n, A_{k+1} = j | M_{k-1} = n, A_k = i\} = p_{ij} \frac{\Pr\{M_{k-1} = n, M_k = n, A_k = i\}}{\Pr\{M_{k-1} = n, A_k = i\}}$$

Since

$$\begin{aligned}
\Pr\{M_k = n, A_k = i, M_{k-1} = n\} &= \sum_{l=1}^{m} \Pr\{M_k = n, A_k = i; M_{k-1} = n, A_{k-1} = l\} \\
&= \sum_{l=1}^{m} \Pr\{M_k = n, A_k = i | M_{k-1} = n, A_{k-1} = l\} \\
&\quad \times \Pr\{M_{k-1} = n, A_{k-1} = l\} \\
&= \sum_{l=1}^{m} [1 - (N-n)\theta_i] p_{li} \Pr\{M_{k-1} = n, A_{k-1} = l\} \\
&= [1 - (N-n)\theta_i] \sum_{l=1}^{m} p_{li} \Pr\{M_{k-1} = n, A_{k-1} = l\}
\end{aligned}$$

and

$$\begin{aligned}
\Pr\{M_{k-1} = n, A_k = i\} &= \sum_{l=1}^{m} \Pr\{M_{k-1} = n, A_k = i, A_{k-1} = l\} \\
&= \sum_{l=1}^{m} \Pr\{A_k = i | M_{k-1} = n, A_{k-1} = l\} \Pr\{M_{k-1} = n, A_{k-1} = l\} \\
&= \sum_{l=1}^{m} \Pr\{A_k = i | A_{k-1} = l\} \Pr\{M_{k-1} = n, A_{k-1} = l\} \\
&= \sum_{l=1}^{m} p_{li} \Pr\{M_{k-1} = n, A_{k-1} = l\}
\end{aligned}$$

Therefore

$$\Pr\{M_k = n, A_{k+1} = j | M_{k-1} = n, A_k = i\} = [1 - (N-n)\theta_i] p_{ij}$$

Similarly, we arrive at

$$\Pr\{M_k = n+1, A_{k+1} = j | M_{k-1} = n, A_k = i\} = (N-n)\theta_i p_{ij}$$

□

Proof of Proposition 4.2. We have

$$\begin{aligned}
E_{\langle M(0), A(0+)\rangle} f(M(t), A(t)) &= E_{\langle M(0), A(0+)\rangle} f(M_{H(t)}, A_{H(t)}) \\
&= \sum_{k=0}^{\infty} E_{\langle M(0), A(0+)\rangle} f(M_{H(t)}, A_{H(t)}) I_{\{h_k < t \le h_{k+1}\}} \\
&= \sum_{k=0}^{\infty} E_{\langle M(0), A(0+)\rangle} f(M_{H(t)}, A_{H(t)}) I_{\{h_k < t < h_{k+1}\}} \\
&\quad + \sum_{k=0}^{\infty} E_{\langle M(0), A(0+)\rangle} f(M_{H(t)}, A_{H(t)}) I_{\{h_k < t = h_{k+1}\}}
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{k=0}^{\infty} E_{\langle M(0),A(0+)\rangle} f(M_k, A_{k+1}) I_{\{h_k<t<h_{k+1}\}} \\
&\quad + \sum_{k=0}^{\infty} E_{\langle M(0),A(0+)\rangle} f(M_{k+1}, A_{k+1}) I_{\{h_k<t=h_{k+1}\}} \\
&= \sum_{k=0}^{\infty} E_{\langle M(0),A(0+)\rangle} f(M_k, A_{k+1}) \Pr\{h_k < t < h_{k+1}\}
\end{aligned}
$$

Note

$$
\Pr\{h_k < t < h_{k+1}\} = 1 - \sum_{l=1}^{k-1} \mathrm{e}^{-\lambda t} \frac{(\lambda t)^l}{l!} - 1 + \sum_{l=1}^{k} \mathrm{e}^{-\lambda t} \frac{(\lambda t)^l}{l!} = \mathrm{e}^{-\lambda t} \frac{(\lambda t)^k}{k!}
$$

Therefore

$$
E_{\langle M(0),A(0+)\rangle} f(M(t), A(t)) = \sum_{k=0}^{\infty} E_{\langle M(0),A(0+)\rangle} f(M_k, A_{k+1}) \mathrm{e}^{-\lambda t} \frac{(\lambda t)^k}{k!}
$$

□

Proof of Proposition 5.1. For any $i_3 > i_2 > i_1 \geq 0$ and $k_3 > k_2 > k_1 \geq 0$, since $\langle M_i, A_i; i = 1, 2, \ldots \rangle$ is a Markov chain, we have

$$
\begin{aligned}
&\Pr\{M_{i_3} = k_3, M_{i_2} = k_2, M_{i_1} = k_1\} \\
&\quad = \sum_{A_{i_3}, A_{i_3-1} \ldots, A_{i_2+1}, A_{i_2}, A_{i_2-1}, \ldots, A_{i_1+1}, A_{i_1}, A_{i_1-1}, \ldots, A_1} \; \sum_{M_{i_3-1}, M_{i_3-2} \ldots, M_{i_2+1}, M_{i_2-1}, M_{i_2-2}, \ldots, M_{i_1+1}, M_{i_1-1}, M_{i_1-2}, \ldots, M_1, M_0} \\
&\Pr\Big\{ M_{i_3} = k_3, A_{i_3}; M_{i_3-1}, A_{i_3-1}; \ldots; M_{i_2+1}, A_{i_2+1}; M_{i_2} = k_2, A_{i_2}; M_{i_2-1}, A_{i_2-1}; \ldots; \\
&\qquad M_{i_1+1}, A_{i_1+1}; M_{i_1} = k_1, A_{i_1}; M_{i_1-1}, A_{i_1-1}; \ldots; M_1, A_1; M_0 \Big\} \\
&\quad = \sum_{A_{i_3}, A_{i_3-1} \ldots, A_{i_2+1}, A_{i_2}, A_{i_2-1}, \ldots, A_{i_1+1}, A_{i_1}, A_{i_1-1}, \ldots, A_1} \; \sum_{M_{i_3-1}, M_{i_3-2} \ldots, M_{i_2+1}, M_{i_2-1}, M_{i_2-2}, \ldots, M_{i_1+1}, M_{i_1-1}, M_{i_1-2}, \ldots, M_1, M_0}
\end{aligned}
$$

$$
\begin{aligned}
&\Pr\{M_{i_3} = k_3, A_{i_3} | M_{i_3-1}, A_{i_3-1}\} \Pr\{M_{i_3-1}, A_{i_3-1} | M_{i_3-2}, A_{i_3-2}\} \ldots \\
&\Pr\{M_{i_2+1}, A_{i_2+1} | M_{i_2} = k_2, A_{i_2}\} \ldots \Pr\{M_1, A_1 | M_0\} \\
&\quad = \sum_{A_{i_3}, A_{i_3-1} \ldots, A_{i_2+1}} \; \sum_{A_{i_2}, A_{i_2-1}, \ldots, A_1} \; \sum_{M_{i_3-1}, M_{i_3-2} \ldots, M_{i_2+1}} \; \sum_{M_{i_2-1}, M_{i_2-2}, \ldots, M_0}
\end{aligned}
$$

$$
\begin{aligned}
&\Pr\{M_{i_3} = k_3, A_{i_3} | M_{i_3-1}, A_{i_3-1}\} \Pr\{M_{i_3-1}, A_{i_3-1} | M_{i_3-2}, A_{i_3-2}\} \ldots \\
&\Pr\{M_{i_2+1}, A_{i_2+1} | M_{i_2} = k_2, A_{i_2}\} \ldots \Pr\{M_1, A_1 | M_0\} \\
&\quad = \sum_{A_{i_3}, A_{i_3-1} \ldots, A_{i_2+1}} \sum_{M_{i_3-1}, M_{i_3-2} \ldots, M_{i_2+1}} \Pr\{M_{i_3} = k_3, A_{i_3} | M_{i_3-1}, A_{i_3-1}\} \ldots \\
&\qquad \times \Pr\{M_{i_2+2}, A_{i_2+2} | M_{i_2+1}, A_{i_2+1}\} \\
&\qquad \times \sum_{A_{i_2}} \Pr\{M_{i_2+1}, A_{i_2+1} | M_{i_2} = k_2, A_{i_2}\} \\
&\qquad \times \sum_{A_{i_2-1}, \ldots, A_1} \sum_{M_{i_2-1}, M_{i_2-2}, \ldots, M_0} \Pr\{M_{i_2} = k_2, A_{i_2} | M_{i_2-1}, A_{i_2-1}\} \ldots \Pr\{M_1, A_1 | M_0\}
\end{aligned}
$$

Since $p_{ij} \equiv p_j$; $\quad i, j = 1, 2, \ldots, m$, we have

$$
\begin{aligned}
\Pr\{M_{i_2+1}, A_{i_2+1} | M_{i_2} = k_2, A_{i_2}\} &= \sum_{k=k_2}^{k_2+1} \Pr\{M_{i_2+1} = k, A_{i_2+1} | M_{i_2} = k_2, A_{i_2}\} \\
&= \sum_{k=k_2}^{k_2+1} \sum_{j=1}^{m} \Pr\{M_{i_2+1} = k, A_{i_2+1} = j | M_{i_2} = k_2, A_{i_2}\} \\
&= \sum_{j=1}^{m} \Pr\{M_{i_2+1} = k_2, A_{i_2+1} = j | M_{i_2} = k_2, A_{i_2}\} \\
&\quad + \sum_{j=1}^{m} \Pr\{M_{i_2+1} = k_2 + 1, A_{i_2+1} = j | M_{i_2} = k_2, A_{i_2}\} \\
&= \sum_{j=1}^{m} [1 - (N - k_2)\theta_j] p_{A_{i_2} j} + \sum_{j=1}^{m} (N - k_2)\theta_j p_{A_{i_2} j} \\
&= \sum_{j=1}^{m} [1 - (N - k_2)\theta_j] p_j + \sum_{j=1}^{m} (N - k_2)\theta_j p_j
\end{aligned}
$$

Therefore

$$
\begin{aligned}
&\Pr\{M_{i_3} = k_3, M_{i_2} = k_2, M_{i_1} = k_1\} \\
&\quad = \sum_{A_{i_3}, A_{i_3-1} \ldots, A_{i_2+2} M_{i_3-1}, M_{i_3-2} \ldots, M_{i_2+2}} \Pr\{M_{i_3} = k_3, A_{i_3} | M_{i_3-1}, A_{i_3-1}\} \ldots \\
&\qquad \times \Pr\{M_{i_2+3}, A_{i_2+3} | M_{i_2+2}, A_{i_2+2}\} \sum_{A_{i_2}} \sum_{A_{i_2+1}=1}^{m} \sum_{k=k_2}^{k_2+1} \Pr\{M_{i_2+2}, A_{i_2+2} | M_{i_2+1} = k, A_{i_2+1}\}
\end{aligned}
$$

$$\Pr\{M_{i_2+1}, A_{i_2+1} | M_{i_2} = k_2, A_{i_2}\} \sum_{A_{i_2-1}, \ldots, A_1} \sum_{M_{i_2-1}, M_{i_2-2}, \ldots, M_0} \Pr\{M_{i_2} = k_2, A_{i_2} | M_{i_2-1}, A_{i_2-1}\} \ldots$$
$$\times \Pr\{M_1, A_1 | M_0\}$$
$$= \sum_{A_{i_3}, A_{i_3-1} \ldots, A_{i_2+2} M_{i_3-1}} \sum_{M_{i_3-2} \ldots, M_{i_2+2}} \Pr\{M_{i_3} = k_3, A_{i_3} | M_{i_3-1}, A_{i_3-1}\} \ldots$$
$$\times \Pr\{M_{i_2+3}, A_{i_2+3} | M_{i_2+2}, A_{i_2+2}\}$$
$$\times \sum_{A_{i_2}} \left[\sum_{A_{i_2+1}=1}^{m} \Pr\{M_{i_2+2}, A_{i_2+2} | M_{i_2+1} = k_2, A_{i_2+1}\} \left[1 - (N - k_2)\theta_{A_{i_2+1}}\right] p_{A_{i_2+1}} \right.$$
$$\left. + \sum_{A_{i_2+1}=1}^{m} \Pr\{M_{i_2+2}, A_{i_2+2} | M_{i_2+1} = k_2 + 1, A_{i_2+1}\} (N - k_2)\theta_{A_{i_2+1}} p_{A_{i_2+1}} \right]$$
$$\times \sum_{A_{i_2-1}, \ldots, A_1} \sum_{M_{i_2-1}, M_{i_2-2}, \ldots, M_0}$$

$$\Pr\{M_{i_2} = k_2, A_{i_2} | M_{i_2-1}, A_{i_2-1}\} \ldots \Pr\{M_1, A_1 | M_0\}$$
$$= \sum_{A_{i_3}, A_{i_3-1} \ldots, A_{i_2+2}, A_{i_2+1}} \sum_{M_{i_3-1}, M_{i_3-2} \ldots, M_{i_2+2}, M_{i_2+1}} \left[\Pr\{M_{i_3} = k_3, A_{i_3} | M_{i_3-1}, A_{i_3-1}\} \times \cdots \right.$$
$$\left. \times \Pr\{M_{i_2+3}, A_{i_2+3} | M_{i_2+2}, A_{i_2+2}\} \Pr\{M_{i_2+1}, A_{i_2+1} | M_{i_2} = k_2\}\right]$$
$$\times \sum_{A_{i_2}, A_{i_2-1}, \ldots, A_1} \sum_{M_{i_2-1}, M_{i_2-2}, \ldots, M_0} \Pr\{M_{i_2} = k_2, A_{i_2} | M_{i_2-1}, A_{i_2-1}\} \ldots \Pr\{M_1, A_1 | M_0\}$$

This implies

$$\Pr\{M_{i_3} = k_3, M_{i_2} = k_2, M_{i_1} = k_1\} = \Pr\{M_{i_3} = k_3 | M_{i_2} = k_2\} \Pr\{M_{i_2} = k_2, M_{i_1} = k_1\}$$
$$= \Pr\{M_{i_3+l-i_2} = k_3 | M_l = k_2\} \Pr\{M_{i_2} = k_2, M_{i_1} = k_1\}$$

where $l \geq k_2$. That is, $\{M_i; i = 0, 1, 2, \ldots\}$ is a time-homogeneous Markov chain. □

Proof of Proposition 5.2. Let

$$\Delta^{\langle c,l1\rangle\langle b,j\rangle}_{\langle c,l\rangle\langle b,j\rangle}(s,t) = P_{\langle c,l\rangle\langle b,j\rangle}(s,t) - P_{\langle c,l1\rangle\langle b,j\rangle}(s,t)$$

Since both $P_{\langle c,l\rangle\langle b,j\rangle}(s,t)$ and $P_{\langle c,l1\rangle\langle b,j\rangle}(s,t)$ satisfy the system of backward differential equations presented in this subsection and the same initial condition (2), we have

$$\Delta^{\langle c,l'\rangle\langle b,j\rangle}_{\langle c,l\rangle\langle b,j\rangle}(s,t) = 0$$

This is due to that the system of backward differential equation has a unique solution.

Further we note that $\{\langle M(t), A(t)\rangle; t \geq 0\}$ is a Markov process. We obtain

$$\Pr\{M(t) = b_1, A(t) = j_1, M(t+u) = b_2, A(t+u) = j_2 | M(s) = c, A(s) = l\}$$
$$= \Pr\{M(t+u) = b_2, A(t+u) = j_2 | M(t) = b_1, A(t) = j_1, M(s) = c, A(s) = l\}$$
$$\times \Pr\{M(t) = b_1, A(t) = j_1 | M(s) = c, A(s) = l\}$$
$$= \Pr\{M(t+u) = b_2, A(t+u) = j_2 | M(t) = b_1, A(t) = j_1\}$$
$$\times \Pr\{M(t) = b_1, A(t) = j_1 | M(s) = c, A(s) = l\}$$
$$= P_{\langle b_1,j_1\rangle\langle b_2,j_2\rangle}(t, t+u) \times P_{\langle c,l\rangle\langle b_1,j_1\rangle}(s,t)$$

Similarly, we obtain

$$\begin{aligned}&\Pr\{M(t)=b_1,A(t)=j_1,M(t+u)=b_2,A(t+u)=j_2|M(s)=c,A(s)=l1\}\\&\quad=P_{\langle b_1,j_1\rangle\langle b_2,j_2\rangle}(t,t+u)\times P_{\langle c,l1\rangle\langle b_1,j_1\rangle}(s,t)\end{aligned}$$

Therefore

$$\begin{aligned}&\Pr\{M(t)=b_1,A(t)=j_1,M(t+u)=b_2,A(t+u)=j_2|M(s)=c,A(s)=l\}\\&\quad=\Pr\{M(t)=b_1,A(t)=j_1,M(t+u)=b_2,A(t+u)=j_2|M(s)=c,A(s)=l1\}\end{aligned}$$

The Markovian property of $\{\langle M(t),A(t)\rangle;t\geq 0\}$ implies that $P_{\langle c,l\rangle}$ and $P_{\langle c,l1\rangle}$ have the same finite dimensional distributions. Then from the monotone theorem of the measure theory (Revuz and Yor 1991) we conclude that the two probability measures $P_{\langle c,l\rangle}$ and $P_{\langle c,l1\rangle}$ are equal. □

Proof of Proposition 5.4. First for any $0<t_0<t_1$, we have

$$\begin{aligned}&\Pr\{M(t_0)=1,M(t_1)-M(t_0)=0\}=\Pr\{M(t_0)=1,M(t_1)=1\}\\&=\sum_{k=0}^{\infty}\sum_{l=k}^{\infty}\Pr\{M_k=1,M_l=1\}\Pr\{H(t_0)=k,H(t_1)=l\}\\&=\sum_{k=0}^{\infty}\sum_{l=k}^{\infty}\Pr\{M_k=1,M_l=1\}\Pr\{H(t_0)=k,H(t_1)-H(t_0)=l-k\}\\&=\sum_{k=0}^{\infty}\sum_{l=k}^{\infty}\Pr\{M_k=1,M_l=1\}\mathrm{e}^{-\lambda t_0}\frac{(\lambda t_0)^k}{k!}\mathrm{e}^{-\lambda(t_1-t_0)}\frac{[\lambda(t_1-t_0)]^{l-k}}{(l-k)!}\\&=\sum_{k=0}^{\infty}\sum_{l=0}^{\infty}\Pr\{M_k=1,M_{l+k}=1\}\mathrm{e}^{-\lambda t_0}\frac{(\lambda t_0)^k}{k!}\mathrm{e}^{-\lambda(t_1-t_0)}\frac{[\lambda(t_1-t_0)]^{l}}{l!}\\&=\sum_{k=0}^{\infty}\sum_{l=0}^{\infty}\Pr\{M_{l+k}=1|M_k=1\}\Pr\{M_k=1\}\mathrm{e}^{-\lambda t_0}\frac{(\lambda t_0)^k}{k!}\mathrm{e}^{-\lambda(t_1-t_0)}\frac{[\lambda(t_1-t_0)]^{l}}{l!}\\&=\sum_{k=0}^{\infty}\sum_{l=0}^{\infty}[1-(N-1)\theta]^l\Pr\{M_k=1\}\mathrm{e}^{-\lambda t_0}\frac{(\lambda t_0)^k}{k!}\mathrm{e}^{-\lambda(t_1-t_0)}\frac{[\lambda(t_1-t_0)]^{l}}{l!}\\&=\sum_{k=0}^{\infty}\Pr\{M_k=1\}\mathrm{e}^{-\lambda t_0}\frac{(\lambda t_0)^k}{k!}\sum_{l=0}^{\infty}[1-(N-1)\theta]^l\mathrm{e}^{-\lambda(t_1-t_0)}\frac{[\lambda(t_1-t_0)]^{l}}{l!}\\&=\sum_{k=0}^{\infty}\Pr\{M_k=1\}\mathrm{e}^{-\lambda t_0}\frac{(\lambda t_0)^k}{k!}\mathrm{e}^{-\lambda(t_1-t_0)}\mathrm{e}^{[1-(N-1)\theta][\lambda(t_1-t_0)]}\\&=\mathrm{e}^{-\lambda(N-1)\theta(t_1-t_0)}\sum_{k=0}^{\infty}\Pr\{M_k=1\}\mathrm{e}^{-\lambda t_0}\frac{(\lambda t_0)^k}{k!}\end{aligned}$$

On the other hand, it holds

$$\Pr\{M(t_0)=1\}=\sum_{k=0}^{\infty}\Pr\{M_k=1\}\mathrm{e}^{-\lambda t_0}\frac{(\lambda t_0)^k}{k!}$$

and

$$\begin{aligned}
\Pr\{M(t_1)-M(t_0)=0\} &= \sum_{k=0}^{\infty}\sum_{l=k}^{\infty}\Pr\{M_l-M_k=0\}\Pr\{H(t_0)=k,H(t_1)=l\} \\
&= \sum_{k=0}^{\infty}\sum_{l=0}^{\infty}\Pr\{M_{l+k}=M_k\}\mathrm{e}^{-\lambda t_0}\frac{(\lambda t_0)^k}{k!}\mathrm{e}^{-\lambda(t_1-t_0)}\frac{[\lambda(t_1-t_0)]^l}{l!} \\
&= \sum_{n=0}^{N}\sum_{k=0}^{\infty}\sum_{l=0}^{\infty}\Pr\{M_{l+k}=M_k=n\}\mathrm{e}^{-\lambda t_0}\frac{(\lambda t_0)^k}{k!}\mathrm{e}^{-\lambda(t_1-t_0)}\frac{[\lambda(t_1-t_0)]^l}{l!} \\
&= \sum_{n=0}^{N}\sum_{k=0}^{\infty}\sum_{l=0}^{\infty}[1-(N-n)\theta]^l\Pr\{M_k=n\}\mathrm{e}^{-\lambda t_0}\frac{(\lambda t_0)^k}{k!}\mathrm{e}^{-\lambda(t_1-t_0)}\frac{[\lambda(t_1-t_0)]^l}{l!} \\
&= \sum_{n=0}^{N}\sum_{k=0}^{\infty}\Pr\{M_k=n\}\mathrm{e}^{-\lambda t_0}\frac{(\lambda t_0)^k}{k!}\sum_{l=0}^{\infty}\mathrm{e}^{-\lambda(t_1-t_0)}\frac{[\lambda(t_1-t_0)]^l[1-(N-n)\theta]^l}{l!} \\
&= \sum_{n=0}^{N}\sum_{k=0}^{\infty}\Pr\{M_k=n\}\mathrm{e}^{-\lambda t_0}\frac{(\lambda t_0)^k}{k!}\mathrm{e}^{-\lambda(t_1-t_0)}\mathrm{e}^{\lambda[1-(N-n)\theta](t_1-t_0)} \\
&= \sum_{n=0}^{N}\sum_{k=0}^{\infty}\Pr\{M_k=n\}\mathrm{e}^{-\lambda t_0}\frac{(\lambda t_0)^k}{k!}\mathrm{e}^{-\lambda(N-n)\theta(t_1-t_0)}
\end{aligned}$$

In this way

$$\begin{aligned}
&\Pr\{M(t_0)=1\}\Pr\{M(t_1)-M(t_0)=0\} \\
&= \left[\sum_{k=0}^{\infty}\Pr\{M_k=1\}\mathrm{e}^{-\lambda t_0}\frac{(\lambda t_0)^k}{k!}\right]\left[\sum_{n=0}^{N}\sum_{k=0}^{\infty}\Pr\{M_k=n\}\mathrm{e}^{-\lambda t_0}\frac{(\lambda t_0)^k}{k!}\mathrm{e}^{-\lambda(N-n)\theta(t_1-t_0)}\right] \\
&> \left[\sum_{k=0}^{\infty}\Pr\{M_k=1\}\mathrm{e}^{-\lambda t_0}\frac{(\lambda t_0)^k}{k!}\right]\left[\sum_{n=0}^{N}\sum_{k=0}^{\infty}\Pr\{M_k=n\}\mathrm{e}^{-\lambda t_0}\frac{(\lambda t_0)^k}{k!}\mathrm{e}^{-\lambda(N-1)\theta(t_1-t_0)}\right] \\
&= \left[\sum_{k=0}^{\infty}\Pr\{M_k=1\}\mathrm{e}^{-\lambda t_0}\frac{(\lambda t_0)^k}{k!}\right]\left[\sum_{k=0}^{\infty}\mathrm{e}^{-\lambda t_0}\frac{(\lambda t_0)^k}{k!}\mathrm{e}^{-\lambda(N-1)\theta(t_1-t_0)}\right] \\
&= \mathrm{e}^{-\lambda(N-1)\theta(t_1-t_0)}\sum_{k=0}^{\infty}\Pr\{M_k=1\}\mathrm{e}^{-\lambda t_0}\frac{(\lambda t_0)^k}{k!} \\
&= \Pr\{M(t_0)=1, M(t_1)-M(t_0)=0\}
\end{aligned}$$

This implies that $\{M(t); t \geq 0\}$ is not an independent increment process. □

Proof of Proposition 5.5. First we note that from Corollary 5.1, $\{M(t); t \geq 0\}$ is a time-homogeneous Markov process. Then from Section 4 and the Kolmogorov-Chapman equation of $\{\langle M(t), A(t)\rangle; t \geq 0\}$ we can obtain the system of backward differential equations for the process as follows

$$\begin{cases}
\frac{\mathrm{d}P_{\langle 0,l\rangle\langle 0,j\rangle}(s,t)}{\mathrm{d}t} = -\lambda P_{\langle 0,l\rangle\langle 0,j\rangle}(s,t) + \lambda\sum_{k=1}^{m} P_{K\langle 0,l\rangle\langle 0,k\rangle}(s,t)[1-N\theta]p_{kj} \\
\frac{\mathrm{d}P_{\langle 0,l\rangle\langle b,j\rangle}(s,t)}{\mathrm{d}t} = -\lambda P_{\langle 0,l\rangle\langle b,j\rangle}(s,t) + \lambda\sum_{k=1}^{m} P_{\langle 0,l\rangle\langle b,k\rangle}(s,t)[1-(N-b)\theta]p_{kj} \\
\qquad\qquad +\lambda\sum_{k=1}^{m} P_{\langle 0,l\rangle\langle b-1,k\rangle}(s,t)[N-b+1]\theta p_{kj}; \quad 1 \leq b < N \\
\frac{\mathrm{d}P_{\langle 0,l\rangle\langle b,j\rangle}(s,t)}{\mathrm{d}t} = \lambda\sum_{k=1}^{m} P_{\langle 0,l\rangle\langle N-1,k\rangle}(s,t)\theta p_{kj}; \quad c < N
\end{cases}$$

where $l, j \in \{1, 2, \ldots, m\}$, and we suppose that at time s the process has $M(s) = 0$, $A(s+) = l$.

For the first equation above we can have the following as a result of summation across $j = 1, 2, \ldots, m$,

$$\frac{\mathrm{d}P_{\langle 0,l\rangle\langle 0\rangle}(s,t)}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\sum_{j=1}^{m} P_{\langle 0,l\rangle\langle 0,j\rangle}(s,t) = -\lambda P_{\langle 0,l\rangle\langle 0\rangle}(s,t) + \lambda\sum_{j=1}^{m}\sum_{k=1}^{m} P_{\langle 0,l\rangle\langle 0,k\rangle}(s,t)[1 - N\theta]p_{kj}$$

$$= -N\theta\lambda P_{\langle 0,l\rangle\langle 0\rangle}(s,t)$$

where we denote

$$P_{\langle 0,l\rangle\langle b\rangle}(s,t) = \sum_{j=1}^{m} P_{\langle 0,l\rangle\langle b,j\rangle}(s,t)$$

for $b = 0, 1, \ldots, N$. Similarly, for $b = 1, 2, \ldots, N-1$ we can obtain

$$\begin{aligned}
\frac{\mathrm{d}P_{\langle 0,l\rangle\langle b\rangle}(s,t)}{\mathrm{d}t} &= \frac{\mathrm{d}}{\mathrm{d}t}\sum_{j=1}^{m} P_{\langle 0,l\rangle\langle b,j\rangle}(s,t) \\
&= -\lambda P_{\langle 0,l\rangle\langle b\rangle}(s,t) + \lambda\sum_{j=1}^{m}\sum_{k=1}^{m} P_{\langle 0,l\rangle\langle b,k\rangle}(s,t)[1 - (N-b)\theta]p_{kj} \\
&\quad + \lambda\sum_{j=1}^{m}\sum_{k=1}^{m} P_{\langle 0,l\rangle\langle b-1,k\rangle}(s,t)[N - b + 1]\theta p_{kj} \\
&= -\lambda P_{\langle 0,l\rangle\langle b\rangle}(s,t) + \lambda[1 - (N-b)\theta]P_{\langle 0,l\rangle\langle b\rangle}(s,t) \\
&\quad + \lambda[N - b + 1]\theta P_{\langle 0,l\rangle\langle b-1\rangle}(s,t) \\
&= -\lambda(N-b)\theta P_{\langle 0,l\rangle\langle b\rangle}(s,t) \\
&\quad + \lambda[N - b + 1]\theta P_{\langle 0,l\rangle\langle b-1\rangle}(s,t)
\end{aligned}$$

For $b = N$, we have

$$\frac{\mathrm{d}P_{\langle 0,l\rangle\langle N\rangle}(s,t)}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\sum_{j=1}^{m} P_{\langle 0,l\rangle\langle N,j\rangle}(s,t) = \lambda\sum_{j=1}^{m}\sum_{k=1}^{m} P_{\langle 0,l\rangle\langle N-1,k\rangle}(s,t)\theta p_{kj} = \lambda\theta P_{\langle 0,l\rangle\langle N-1\rangle}(s,t)$$

Therefore the corresponding Q-matrix for $\{M(t); t \geq 0\}$ is as follows

$$\begin{bmatrix}
-N\theta\lambda & N\theta\lambda & 0 & & & \\
0 & -(N-1)\theta\lambda & (N-1)\theta\lambda & & & \\
 & & & \ddots & & \\
 & & & & -\theta\lambda & \theta\lambda \\
 & & & & 0 & 0
\end{bmatrix}$$

Similar to Proposition 5.3, we can claim that

$$\Pr\{\tau_1 > t | M(0)\} = \mathrm{e}^{-N\theta\lambda t}$$

$$\Pr\{\tau_i > t | M(0)\} = \mathrm{e}^{-(N-i)\theta\lambda t}; \quad i = 2, 3, \ldots, N-1$$

□

Proof of Proposition 7.1. Denote $\alpha_i(k) = \Pr\{M_i = k\}$; $k = 0, 1, \ldots, N-1$; $i = 1, 2, \ldots$. We have

$$\begin{aligned}
\alpha_{i+1}(k+1) &= \Pr\{M_{i+1} = k+1, M_i = k+1\} + \Pr\{M_{i+1} = k+1, M_i = k\} \\
&= \sum_{j=1}^{m}\sum_{l=1}^{m} \Pr\{M_{i+1} = k+1, A_{i+1} = j, M_i = k+1, A_i = l\} \\
&\quad + \sum_{j=1}^{m}\sum_{l=1}^{m} \Pr\{M_{i+1} = k+1, A_{i+1} = j, M_i = k, A_i = l\} \\
&= \sum_{j=1}^{m}\sum_{l=1}^{m} [1 - (N-k-1)\theta_j] p_{lj} \Pr\{M_i = k+1, A_i = l\} \\
&\quad + \sum_{j=1}^{m}\sum_{l=1}^{m} (N-k)\theta_j p_{lj} \Pr\{M_i = k, A_i = l\} \\
&= \sum_{j=1}^{m} [1 - (N-k-1)\theta_j] p_j \Pr\{M_i = k+1\} + \sum_{j=1}^{m} (N-k)\theta_j p_j \Pr\{M_i = k\} \\
&= \sum_{j=1}^{m} [1 - (N-k-1)\theta_j] p_j \alpha_i(k+1) + \sum_{j=1}^{m} (N-k)\theta_j p_j \alpha_i(k)
\end{aligned}$$

Let

$$f(k) = \sum_{j=1}^{m} [1 - (N-k)\theta_j] p_j$$

$$g(k) = \sum_{j=1}^{m} (N-k+1)\theta_j p_j$$

Then

$$\alpha_{i+1}(k+1) = f(k+1)\alpha_i(k+1) + g(k+1)\alpha_i(k)$$

with the boundary condition

$$\alpha_1(0) = \Pr\{M_1 = 0\} = \sum_{j=1}^{m} (1 - N\theta_j) p_j$$

$$\alpha_1(1) = \Pr\{M_1 = 1\} = \sum_{j=1}^{m} N\theta_j p_j$$

$$\begin{aligned}
\alpha_{i+1}(0) &= \Pr\{M_1 = \cdots M_i = M_{i+1} = 0\} \\
&= \sum_{j=1}^{m} (1 - N\theta_j) p_j \Pr\{M_i = 0\} = f(0)\alpha_i(0); \quad i = 1, 2, \ldots
\end{aligned}$$

Now we note that

$$EM_{i+1} = \sum_{k=1}^{N} k \Pr\{M_{i+1} = k\} = \sum_{k=1}^{N} k\alpha_{i+1}(k) = \sum_{k=1}^{N} kf(k)\alpha_i(k) + \sum_{k=1}^{N} kg(k)\alpha_i(k-1)$$

$$= \sum_{k=1}^{N} kf(k)\alpha_i(k) + \sum_{k=0}^{N-1} (k+1)g(k+1)\alpha_i(k)$$

$$= \sum_{k=1}^{N} kf(k)\alpha_i(k) + \sum_{k=0}^{N-1} kg(k+1)\alpha_i(k) + \sum_{k=0}^{N-1} g(k+1)\alpha_i(k)$$

$$= \sum_{k=1}^{N} kf(k)\alpha_i(k) + \sum_{k=1}^{N} kg(k+1)\alpha_i(k) - Ng(N+1)\alpha_i(N) + \sum_{k=0}^{N-1} g(k+1)\alpha_i(k)$$

$$= \sum_{k=1}^{N} k(f(k) + g(k+1))\alpha_i(k) + \sum_{k=0}^{N-1} g(k+1)\alpha_i(k)$$

$$= \sum_{k=1}^{N} k\alpha_i(k) + \sum_{k=0}^{N-1} g(k+1)\alpha_i(k)$$

$$= EM_i + \sum_{k=0}^{N-1} g(k+1)\alpha_i(k)$$

In above derivation we have exploited the fact that

$$f(k) + g(k+1) \equiv 1$$

Since

$$g(k+1) = (N-k)\hat{\theta}$$

We obtain

$$EM_{i+1} = EM_i + \sum_{k=0}^{N-1} (N-k)\hat{\theta}\alpha_i(k)$$

$$= EM_i + \sum_{k=0}^{N} (N-k)\hat{\theta}\alpha_i(k)$$

$$= EM_i + \hat{\theta}\sum_{k=0}^{N} N\alpha_i(k) - \hat{\theta}\sum_{k=0}^{N} k\alpha_i(k)$$

$$= EM_i + \hat{\theta}N - \hat{\theta}EM_i$$

$$= (1-\hat{\theta})EM_i + \hat{\theta}N$$

$$= (1-\hat{\theta})[(1-\hat{\theta})EM_{i-1} + \hat{\theta}N] + \hat{\theta}N$$

In this way

$$\begin{aligned}
EM_{i+1} &= (1-\hat{\theta})^i \sum_{j=1}^{m} N\theta_j p_j + \sum_{k=0}^{i-1} (1-\hat{\theta})^k \alpha N \\
&= (1-\hat{\theta})^i \sum_{j=1}^{m} N\theta_j p_j + \hat{\theta} N \frac{1-(1-\hat{\theta})^i}{1-(1-\hat{\theta})} \\
&= (1-\hat{\theta})^i \sum_{j=1}^{m} N\theta_j p_j + N(1-(1-\hat{\theta})^i) = N + \left[\sum_{j=1}^{m} N\theta_j p_j - N\right](1-\hat{\theta})^i \\
&= N + [N\hat{\theta} - N](1-\hat{\theta})^i \\
&= N - N(1-\hat{\theta})^{i+1} \\
&= N - Ne^{-(i+1)\beta}
\end{aligned}$$

Therefore

$$EM_{i+1} = N\lfloor 1 - \mathrm{e}^{(i+1)\beta} \rfloor$$

□

Proof of Proposition 8.3. First from Section 5, we note the corresponding $\{M(t); t \geq 0\}$ is a Markov process. Then for any $i, j \geq 1$, with $i + j \leq N$, it holds

$$E_{M(0)}\lfloor f(\tau_{i+j}) g(\tau_i) \rfloor = E_{M(0)}\lfloor f\left(\vartheta_{t_i} \circ \tau_j\right) g(\tau_i) \rfloor = E_{M(0)} g(\tau_i) \times E_{M(t_i)} f(\tau_j) = E_{M(0)} g(\tau_i) \times E_i f(\tau_j)$$

where f and g are any measurable function defined on $[0, \infty)$, ϑ is the shift operator as defined in the Remarks of Section 5.2, and E_i denotes $E_{M(\cdot)}$ with $M(\cdot) = i$. Since

$$E_{M(0)} f(\tau_{i+j}) = E_{M(0)} f\left(\vartheta_{t_i} \circ \tau_j\right) = E_{M(0)} E_i f(\tau_j) = E_i f(\tau_j)$$

We obtain

$$E_{M(0)}\lfloor f(\tau_{i+j}) g(\tau_i) \rfloor = E_{M(0)} g(\tau_i) E_{M(0)} f(\tau_{i+j})$$

Therefore τ_{i+j} and τ_i are independent with respect to $P_{M(0)}$.

Now we need to prove that for any $1 \leq j_1 < \cdots < j_l \leq N$, it holds that $f_1(\tau_{j_1})$, $f_2(\tau_{j_2}), \ldots, f_l(\tau_{j_l})$ are independent. Note that $f_1(\tau_{j_1})$ is independent of $f_i(\tau_{j_i})$ for each of $i = 2, 3, \ldots, l$, there must hold that $f_1(\tau_{j_1})$ is independent of $f_2(\tau_{j_2}) \times \cdots \times f_l(\tau_{j_l})$. That is,

$$E_{M(0)}\lfloor f_1\left(\tau_{j_1}\right) \times f_2\left(\tau_{j_2}\right) \times \cdots \times f_l\left(\tau_{j_l}\right) \rfloor = E_{M(0)} f_1\left(\tau_{j_1}\right) \times E_{M(0)}\lfloor f_2\left(\tau_{j_2}\right) \times \cdots \times f_l\left(\tau_{j_l}\right) \rfloor$$

Similarly,

$$E_{M(0)}\lfloor f_1\left(\tau_{j_1}\right) \times f_2\left(\tau_{j_2}\right) \times \cdots \times f_l\left(\tau_{j_l}\right) \rfloor = E_{M(0)} f_1\left(\tau_{j_1}\right) \times E_{M(0)} f_2\left(\tau_{j_2}\right) \times \cdots \times E_{M(0)} f_l\left(\tau_{j_l}\right)$$

□

PROPOSITION A.1. For the Markov chain $\{\langle M_i, A_{i+1}\rangle; i = 0, 1, \ldots\}$, denote

$$\beta_{n,j}^k = \Pr\{M_k = n, A_{k+1} = j\}$$

Then it holds

$$\beta_n^k = \beta_n^{k-1} I_n[p_{ij}] + \beta_{n-1}^{k-1} \Im_n[p_{ij}]$$

with $k \geq 1$; $\quad n = 1, 2, \ldots, N-1$, where

$$\beta_n^k = \left[\beta_{n,1}^k, \beta_{n,2}^k, \ldots, \beta_{n,m}^k\right]$$

$$I_n = \begin{bmatrix} 1-(N-n)\theta_1, & 0, & \ldots, & 0 \\ 0 & 1-(N-n)\theta_2, & \ldots, & 0 \\ \vdots & & & \\ 0, & 0, & \ldots, & 1-(N-n)\theta_m \end{bmatrix}$$

$$\Im_n = \begin{bmatrix} (N-n+1)\theta_1, & 0, & \ldots, & 0 \\ 0, & (N-n+1)\theta_2, & \ldots, & 0 \\ \vdots & & & \\ 0, & 0, & \ldots, & (N-n+1)\theta_m \end{bmatrix}$$

The boundary conditions are

$$\beta_{n,j}^0 = \begin{cases} p_j & \text{if } n = 0 \\ 0 & \text{if } n > 0 \end{cases}$$

$$\beta_{n,j}^k \equiv 0 \quad \text{if } k < n$$

$$\beta_0^k = \beta_0^{k-1} \mathrm{I}_0[p_{ij}] \quad \text{for } k > 0$$

$$\beta_N^k = \beta_N^{k-1} + \beta_{N-1}^{k-1} \Im_N[p_{ij}] \quad \text{for } k \geq N$$

Proof. We have

$$\begin{aligned}
\Pr\{M_k = n, A_{k+1} = j\} &= \sum_{M_{k-1}, A_k} \Pr\{M_k = n, A_{k+1} = j; M_{k-1}, A_k\} \\
&= \sum_{i=1}^{m} \Pr\{M_k = n, A_{k+1} = j; M_{k-1} = n, A_k = i\} \\
&\quad + \sum_{i=1}^{m} \Pr\{M_k = n, A_{k+1} = j; M_{k-1} = n-1, A_k = i\} \\
&= \sum_{i=1}^{m} [1 - (N-n)\theta_i] p_{ij} \Pr\{M_{k-1} = n, A_k = i\} \\
&\quad + \sum_{i=1}^{m} (N-n+1)\theta_i p_{ij} \Pr\{M_{k-1} = n-1, A_k = i\} \\
&= \sum_{i=1}^{m} p_{ij} \Pr\{M_{k-1} = n, A_k = i\} \\
&\quad + (N-n) \sum_{i=1}^{m} \theta_i p_{ij} [\Pr\{M_{k-1} = n-1, A_k = i\} \\
&\quad - \Pr\{M_{k-1} = n, A_k = i\}] + \sum_{i=1}^{m} \theta_i p_{ij} \Pr\{M_{k-1} = n-1, A_k = i\}
\end{aligned}$$

Then we have

$$\beta_{n,j}^{k} = \sum_{i=1}^{m} [1 - (N-n)\theta_i] p_{ij} \beta_{n,i}^{k-1} + \sum_{i=1}^{m} (N-n+1)\theta_i p_{ij} \beta_{n-1,i}^{k-1}$$

□

PROPOSITION A.2. For the Markov chain $\{\langle M_i, A_i \rangle; i = 0, 1, \dots\}$, denote

$$\widehat{\beta}_{n,j}^{k} = \Pr\{M_k = n, A_k = j\}$$

Then it holds

$$\widehat{\beta}_{n}^{k} = \widehat{\beta}_{n}^{k-1} [p_{ij}] \mathrm{I}_n + \widehat{\beta}_{n-1}^{k-1} [p_{ij}] \Im_n$$

with $k \geq 2$; $\quad n = 1, 2, \dots, N-1$, where

$$\widehat{\beta}_{n}^{k} = \left[\widehat{\beta}_{n,1}^{k}, \widehat{\beta}_{n,2}^{k}, \dots, \widehat{\beta}_{n,m}^{k} \right]$$

$$I_n = \begin{bmatrix} 1-(N-n)\theta_1, & 0, & \ldots, & 0 \\ 0 & 1-(N-n)\theta_2, & \ldots, & 0 \\ \vdots & & & \\ 0, & 0, & \ldots, & 1-(N-n)\theta_m \end{bmatrix}$$

$$\Im_n = \begin{bmatrix} (N-n+1)\theta_1, & 0, & \ldots, & 0 \\ 0, & (N-n+1)\theta_2, & \ldots, & 0 \\ \vdots & & & \\ 0, & 0, & \ldots, & (N-n+1)\theta_m \end{bmatrix}$$

The boundary conditions are

$$\beta_{n,j}^1 = \begin{cases} 1-N\theta_j & \text{if } n=0 \\ N\theta_j & \text{if } n=1 \\ 0 & \text{if } n>1 \end{cases}$$

$$\beta_{n,j}^k \equiv 0 \quad \text{if } k < n$$

$$\beta_0^k = \beta_0^{k-1} I_0[p_{ij}] \quad \text{for } k > 1$$

$$\beta_N^k = \beta_N^{k-1} + \beta_{N-1}^{k-1} \Im_N[p_{ij}] \quad \text{for } k \geq N$$

Proof. Similar to that of Proposition A.1. □

Kai-Yuan Cai is a Cheung Kong Scholar (Chair Professor), jointly appointed by the Ministry of Education of China and the Li Ka Shing Foundation of Hong Kong in 1999. He has been a full professor at Beihang University (Beijing University of Aeronautics and Astronautics) since 1995. He was born in April 1965 and entered Beihang University as an undergraduate student in 1980. He received his B.S. degree in 1984, M.S. degree in 1987, and Ph.D. degree in 1991, all from Beihang University. He was a research fellow at the Centre for Software Reliability, City University, London, and a visiting scholar at City University of Hong Kong, Swinburge University of Technology (Australia), University of Technology, Sydney (Australia), and Purdue University (USA).

Dr Cai has published over 90 research papers and is the author of three books: *Software Defect and Operational Profile Modeling* (Kluwer, Boston, 1998); *Introduction to Fuzzy Reliability* (Kluwer, Boston, 1996); *Elements of Software Reliability Engineering* (Tshinghua University Press, Beijing, 1995, in Chinese). He serves on the editorial board of the international journal *Fuzzy Sets and Systems* and is the editor of the *Kluwer International Series on Asian Studies in Computer and Information Science* (http://www.wkap.nl/prod/s/ASIS). He served as program committee co-chair for *the Fifth International Conference on Quality Software* (Melbourne, Australia, September 2005), *the First International Workshop on Software Cybernetics* (Hong Kong, September 2004), and *the Second International Workshop on Software Cybernetics* (Edinburgh, UK, July 2005). He also served (will serve) as guest editor for *Fuzzy Sets and Systems* (1996), the *International Journal of Software*

Engineering and Knowledge Engineering (2006), and the *Journal of Systems and Software* (2006). His main research interests include software reliability and testing, intelligent systems and control, and software cybernetics.

Dong Zhao received the M.Sc. degree in Mathematics from Beijing Normal University, China in 1990 and the Ph. D. degree in Probability from the Institute of Applied Mathematics, Academia Sinica, China in 1996. From July 1993 to March 1997, he was an assistant professor in the Institute of Applied Mathematics, Academia Sinica. From September 1997 to September 2002, he was an associated professor in the Institute of Applied Mathematics, Academia Sinica. From September 2002 to now, he was an associated professor in the Academy of Mathematics and Systems Sciences, Academia Sinica. His research interests include stochastic processes, stochastic analysis, stochastic differential equation, Markov processes and potential, Software Reliability.

Ke Liu obtained his B.S. in Applied Mathematics and Operations Research and Ph.D. in Applied Mathematics from the University of Science and Technology of China in 1982 and the University of South Australia in 1997, respectively. Currently, he is a Professor at the Institute of Applied Mathematics, the Academy of Mathematics and System Sciences, the Chinese Academy of Sciences. He was a visiting Professor at City University of Hong Kong, and a visiting scholar at University of Maryland (USA), University of South Australia (Australia), Chinese University (Hong Kong), and University of Science and Technology (Hong Kong).*Dr Liu* has published over 50 research papers and is the author of two books: *Introduction of Markov decision processes* (Liao Ning Ren Min Press, Liao Ning, 1986, in Chinese, co-author with Dong Zeqing) and *Applied Markov Decision processes* (Tshinghua University Press, Beijing, 2004, in Chinese). His research interests are Markov Decision Processes, Stochastic Optimization, Reliability Theory, and Supply Chain Management.

Cheng-Gang Bai received the M.Sc. degree in Statistics from Nanijing University of Aeronautics and Astronautics, Nanjing, China in 1990 and the Ph. D. degree in Control Theory and Control Engineering from Zhejiang University, Zhejiang, China in 1999. From July 1990 to March 1996, he was a Lecturer with the Department of Computer Science and Engineering, Liaocheng Teacher's University, China. From September 1999 to November 2001, he was a Postdoctoral Fellow with the Department of Automatic Control, Beijing University of Aeronautics and Astronautics, China. In November 2001, he joined the faculty of the Department of Automatic Control, Beijing University of Aeronautics and Astronautics, China and has been an Associate Professor since July 2002 in the same university. His research interests include Bayesian Statistics, Software Reliability, Software Testing.