Optimal Testing Resource Allocation, and Sensitivity Analysis in Software Development

Chin-Yu Huang, *Member, IEEE,* and Michael R. Lyu, *Fellow, IEEE*

***Abstract*—We consider two kinds of software testing-resource allocation problems. The first problem is to minimize the number of remaining faults given a fixed amount of testing-effort, and a reliability objective. The second problem is to minimize the amount of testing-effort given the number of remaining faults, and a reliability objective. We have proposed several strategies for module testing to help software project managers solve these problems, and make the best decisions. We provide several systematic solutions based on a nonhomogeneous Poisson process model, allowing systematic allocation of a specified amount of testing-resource expenditures for each software module under some constraints. We describe several numerical examples on the optimal testing-resource allocation problems to show applications & impacts of the proposed strategies during module testing. Experimental results indicate the advantages of the approaches we proposed in guiding software engineers & project managers toward best testing resource allocation in practice. Finally, an extensive sensitivity analysis is presented to investigate the effects of various principal parameters on the optimization problem of testing-resource allocation. The results can help us know which parameters have the most significant influence, and the changes of optimal testing-effort expenditures affected by the variations of fault detection rate & expected initial faults.**

***Index Terms*—Non-homogeneous Poisson processes, sensitivity analysis, software reliability, testing resource allocation.**

ACRONYMS[1]

NHPP	nonhomogeneous Poisson process
SRGM	software reliability growth model
TEF	testing-effort function
TE	testing-effort
MLE	maximum likelihood estimation
LSE	least squares estimation

NOTATION

$m(t)$	expected mean number of faults detected in time $(0,t]$, mean value function
$\lambda(t)$	failure intensity for $m(t)$, $dm(t)/dt$
$w_\kappa(t)$	current testing-effort consumption at time t
$W_\kappa(t)$	cumulative testing-effort consumption at time t
a	expected number of initial faults
r	fault detection rate per unit testing-effort
N	total amount of testing-effort eventually consumed
α	consumption rate of testing-effort expenditures in the generalized *logistic* testing-effort function
A	constant parameter in the generalized *logistic* testing-effort function
κ	structuring index whose value is larger for better structured software development efforts
β	constant parameter
v_i	weighting factor to measure the relative importance of a fault removal from module i
$R(x\|t)$	conditional software reliability

Manuscript received February 1, 2004; revised April 18, 2005. This research was supported by the National Science Council, Taiwan, under Grant NSC 94-2213-E-007-087 and also partially supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project no. CUHK4205/04E). Associate Editor: K. Kanoun.

C.-Y. Huang is with the Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan (e-mail: cyhuang@cs.nthu.edu.tw).

M. R. Lyu is with the Computer Science and Engineering Department, The Chinese University of Hong Kong, Shatin, Hong Kong (e-mail: lyu@cse.cuhk.edu.hk).

Digital Object Identifier 10.1109/TR.2005.858099

[1]The singular and plural of an acronym are always spelled the same.

I. INTRODUCTION

A COMPUTER system comprises two major components: hardware, and software. With the steadily growing power & reliability of hardware, software has been identified as a major stumbling block in achieving desired levels of system dependability. We need quality software to produce, manage, acquire, display, and transmit information anywhere in the world. Software producers must ensure the adequate reliability of the delivered software, the time of delivery, and its cost. According to the ANSI definition: "*Software reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment*" [1]. Alternatively, it may be viewed from the perspective of general use on a variety of different inputs, in which case it is the probability that it will correctly process a randomly chosen input. Many *Software Reliability Growth Models* (SRGM) were developed in the 1970s–2000s [1], [2]. SRGM describe failures as a random process, which is characterized in either times of failures, or the number of failures at fixed times.

In addition to software reliability measurement, SRGM can help us predict the fault detection coverage in the testing phase, and estimate the number of faults remaining in the software systems. From our studies, there are some SRGM that describe the relationship among the calendar testing, the amount of testing-effort, and the number of software faults detected by testing. The testing-effort (TE) can be represented as the number of CPU hours, the number of executed test cases, or human power, etc [2]. Musa *et al.* [2] showed that the effort index, or the execution time is a better time domain for software reliability modeling than the calendar time because the observed reliability growth curve depends strongly on the time distribution of the TE.

In the software development phase, testing begins at the component level, and different testing techniques are appropriate at different points in time. Testing is conducted by the developer of the software, as well as an independent test group [3]. One major software development challenge is that testing is too expensive & lengthy, yet the project schedule has to meet a delivery deadline. Most popular commercial software products are complex systems composed of a number of modules. As soon as the modules are developed, they have to be tested in a variety of ways, and tests are derived from the developer's experience. Practically, module testing is the most detailed form of testing to be performed. Thus, project managers should know how to allocate the specified testing resources among all the modules & develop quality software with high reliability.

From our studies [4]–[23], there are many papers that have addressed the problems of optimal resource allocation. In this paper, we first consider two kinds of software testing-resource allocation problems, and then propose several strategies for module testing. Namely, we provide systematic methods for the software project managers to allocate a specific amount of TE expenditures for each module under some constraints, such as *1) minimizing the number of remaining faults with a reliability objective*, or *2) minimizing the amount of testing-effort with a reliability objective*. Here we employ a SRGM with generalized *logistic* testing-effort function to describe the time-dependency behaviors of detected software faults, and the testing-resource expenditures spent during module testing. The proposed model is based on Non-homogeneous Poisson processes (NHPP).

The remaining contents of this paper consist of four sections. Section II describes an SRGM with generalized *logistic* TEF. In Section III, the methods for testing resource allocation & optimization for modular software testing are introduced. Numerical examples for the optimum TE allocation problems are demonstrated in Section IV. In Section V, we analyze the sensitivity of parameters of proposed SRGM.

II. SRGM WITH GENERALIZED LOGISTIC TESTING-EFFORT FUNCTION

A. *Software Reliability Modeling*

A number of SRGM have been proposed on the subject of software reliability [1]. Traditional SRGM, such as the well-known Goel-Okumoto model, and the Delayed S-shaped model, have been shown to be very useful in fitting software failure data. Yamada *et al.* [6]–[8] modified the G-O model, and incorporated the concept of TE in an NHPP model to get a better description of the software fault phenomenon. Later, Huang *et al.* [24], [25] proposed a new SRGM with the *logistic* TEF to predict the behavior of failure occurrences, and the fault content of a software product. Based on our past experimental results [26], [27], this approach is suitable for estimating the reliability of software application during the development process. The following are the modeling assumptions:

1) The fault removal process is modeled as a NHPP, and the software application is subject to failures at random times caused by the remaining faults in the system.
2) The mean number of faults detected in the time interval $(t, t+\Delta t)$ by the current TE is proportional to the mean number of remaining faults in the system at time t, and the proportionality is a constant over time.
3) TE expenditures are described by a generalized *logistic* TEF.
4) Each time a failure occurs, the corresponding fault is immediately removed, and no new faults are introduced.

Let $m(t)$ be the mean value function of the expected number of faults detected in time $(0, t]$. Because the expected number of detected faults is finite at any time, $m(t)$ is an increasing function of t, and $m(0) = 0$. According to these assumptions, we get

$$\frac{m(t+\Delta t) - m(t)}{w_\kappa(t)} = r \times \big[a - m(t)\big]\Delta t. \tag{1}$$

That is

$$r = \lim_{\Delta t \to 0} \frac{m(t+\Delta t) - m(t)}{\big[a - m(t)\big] w_\kappa(t) \Delta t}. \tag{2}$$

Consequently, if the number of detected faults due to the current TE expenditures is proportional to the number of remaining faults, we obtain the differential equation

$$\frac{dm(t)}{dt} \times \frac{1}{w_\kappa(t)} = r \times \big[a - m(t)\big]. \tag{3}$$

Solving the above differential equation under the boundary condition $m(0) = 0$, we have

$$\begin{aligned} m(t) &= a\Big(1 - \exp\Big[-r\big(W_\kappa(t) - W_\kappa(0)\big)\Big]\Big) \\ &= a\Big(1 - \exp\Big[-r\big(W(t)\big)\Big]\Big). \end{aligned} \tag{4}$$

Note that parameter a is the number of initial faults, and this number is usually a representative measure of software reliability. It can also provide an estimate of the number of failures that will eventually be encountered by the customers. Besides, parameter r is the fault detection rate, or the rate of discovering new faults in software during the testing phase. In general, at the beginning of the testing phase, many faults can be discovered by inspection, and the fault detection rate depends on the fault discovery efficiency, the fault density, the testing-effort, and the inspection rate [3]. In the middle stage of the testing phase, the fault detection rate normally depends on other parameters, such as the execution rate of CPU instruction, the failure-to-fault relationship, the code expansion factor, and the scheduled CPU execution hours per calendar day [2]. We can use this rate to track the progress of checking activities, to evaluate the effectiveness of test planning, and to assess the checking methods we adopted [25]. In fact, $m(t)$ is nondecreasing with respect to testing time t. Knowing its value can help us determine whether the software is ready for release, and if not, how much more of the testing resources are required. It can also provide an estimate of the number of failures that will eventually be encountered by the customers.

Yamada *et al.* [11], [26] reported that the TE could be described by a *Weibull*-type curve, and the *Weibull* curve is one of the three known extreme-value distributions. Although a *Weibull*-type curve can fit the data well under the general software development environment, it will show the apparent peak phenomenon when the value of the shape parameter is greater than 3 [26]. From our past studies [27], [28], a *logistic* TEF with a structuring index was proposed, which can be used to consider & evaluate the effects of possible improvements on software development methodology. The idea of a *logistic* TEF was proposed by F. N. Parr [29]; it predicts essentially the same behavior as the Rayleigh curve, except during the early part of the project. For a sample of some two dozen projects studied in the Yourdon 1978–1980 project survey, the logistic TEF was fairly accurate in describing expended TE [30]. In [28], we extended the *logistic* TEF to a generalized form, and the generalized *logistic* TEF is formulated as

$$W_\kappa(t) = \frac{N}{\sqrt[\kappa]{1 + Ae^{-\alpha\kappa t}}}. \tag{5}$$

The current TE consumption is

$$w_\kappa(t) = \frac{dw_\kappa(t)}{dt}. \tag{6}$$

The TE reaches its maximum value at time

$$t_{\max} = \frac{\ln \frac{A}{\kappa}}{\alpha\kappa}. \tag{7}$$

The conditional reliability function after the last failure occurs at time t is obtained by [1], [2]

$$R_{cond}(t) \equiv R_{cond}(t+\Delta t \mid t) = \exp\left[-\left(m(t+\Delta t) - m(t)\right)\right]. \tag{8}$$

Taking the logarithm on both sides of the above equation, we obtain

$$\ln R_{cond}(t) = -\left(m(t+\Delta t) - m(t)\right). \tag{9}$$

Here we will define another measure of reliability, i.e., the ratio of the cumulative number of detected faults at time t to the expected number of initial faults.

$$R(t) \equiv \frac{m(t)}{a}. \tag{10}$$

Note that $R(t)$ is an increasing function in t. Using $R(t)$, we can obtain the required testing time needed to reach the reliability objective R_0, or decide whether the reliability objective can be satisfied at a specified time. If we know that the value of $R(t)$ has achieved an acceptable level, then we can determine the right time to release this software.

B. Methods of Model's Parameter Estimation

To validate the proposed model, experiments on real software failure data will be performed. Two most popular estimation techniques are *Maximum Likelihood Estimation* (MLE), and *Least Squares Estimation* (LSE) [1], [2], [26]. For example, using the method of LSE, the evaluation formula $S1(N, A, \alpha)$ of (5) with $\kappa = 1$ is depicted as

$$\text{Minimize:} \quad S1(N, A, \alpha) = \sum_{i=1}^{n} \left[W_i^* - W_k(t_i)\right]^2 \tag{11}$$

where W_i^* is the cumulative testing-effort actually consumed in time $(0, t_i]$, and $W_k(t_i)$ is the cumulative TE estimated by (5). Differentiating $S1$ with respect to N, A, and α, setting the partial derivatives to zero, and rearranging these terms, we can solve this type of nonlinear least square problems. We obtain

$$\frac{\partial S1}{\partial N} = \sum_{i=1}^{n} 2\left(W_i^* - \frac{N}{1 + A\exp[-\alpha t]}\right)\frac{1}{1 + A\exp[-\alpha t]} = 0. \tag{12}$$

Thus, the least squares estimator N is given by solving the above equation to yield

$$N = \frac{\sum_{i=1}^{n} 2\left(\frac{W_i^*}{1 + A\exp[-\alpha t]}\right)}{\sum_{i=1}^{n} 2\left(\frac{1}{1 + A\exp[-\alpha t]}\right)^2}. \tag{13}$$

Next, we have

$$\frac{\partial S1}{\partial A} = \sum_{i=1}^{n} 2\left(W_i^* - \frac{N}{1 + A\exp[-\alpha t]}\right)\frac{N\exp[-\alpha t]}{(1 + A\exp[-\alpha t])^2} = 0 \tag{14}$$

and

$$\frac{\partial S1}{\partial \alpha} = \sum_{i=1}^{n} 2\left(W_i^* - \frac{N}{1 + A\exp[-\alpha t]}\right)\frac{NAt\exp[-\alpha t]}{(1 + A\exp[-\alpha t])^2} = 0. \tag{15}$$

The other parameters A & α can also be obtained by substituting the least squares estimator N into (14) & (15). Similarly, if the mean value function is described in (4), then the evaluation formula $S2(a, r)$ can be obtained as

$$\text{Minimize:} \quad S2(a, r) = \sum_{i=1}^{n} \left[m_i^* - m(t_i)\right]^2 \tag{16}$$

where m_i^* is the cumulative number of detected faults in a given time interval $(0, t_i]$, and $m(t_i)$ is the expected number of software faults estimated by (4). Differentiating $S2$ with respect to a & r, setting the partial derivatives to zero, and rearranging these terms, we can solve this type of nonlinear least square problems.

On the other hand, the likelihood function for the parameters a & r in the NHPP model with $m(t)$ in (4) is given by

$$\begin{aligned} L &\equiv P_r\{N(t_1) = m_1, N(t_2) = m_2, \ldots, N(t_n) = m_n\} \\ &= \prod_{i=1}^{n} \frac{\{m(t_i) - m(t_{i-1})\}}{(m_i - m_{i-1})!}\exp\left[-\left(m(t_i) - m(t_{i-1})\right)\right] \end{aligned} \tag{17}$$

where $m_0 \equiv 0$ for t_0. Therefore, taking the logarithm of the likelihood function in (17), we have

$$\begin{aligned} \ln L = &\sum_{i=1}^{n}(m_i - m_{i-1})\ln\left[m(t_i) - m(t_{i-1})\right] \\ &- \sum_{i=1}^{n}\left((m(t_i) - m(t_{i-1})\right) - \sum_{i=1}^{n}\ln\left[(m_i - m_{i-1})!\right]. \end{aligned} \tag{18}$$

From (3), we know that $m(t_i) - m(t_{i-1}) = a(\exp[-rW(t_{i-1})] - \exp[-rW(t_i)])$. Thus,

$$\ln L = \sum_{i=1}^{n}(m_i - m_{i-1})\ln a + \sum_{i=1}^{n}(m_i - m_{i-1}) \times \ln\left[\left(\exp\left[-rW(t_{i-1})\right] - \exp\left[-rW(t_i)\right]\right)\right] - a\left(1 - \exp\left[-rW^*(t_n)\right]\right) - \sum_{i=1}^{n}\ln\left[(m_i^* - m_{i-1}^*)!\right]. \tag{19}$$

Consequently, the maximum likelihood estimates a & r can be obtained by solving

$$\frac{\partial \ln L}{\partial a} = \frac{\partial \ln L}{\partial r} = 0. \tag{20}$$

III. Testing-Resource Allocation for Module Testing

In this section, we will consider several resource allocation problems based on an SRGM with generalized *logistic* TEF during software testing phase.

Assumptions [4], [5], [7], [11]–[14], [27]:

1) The software system is composed of N modules, and the software modules are tested individually. The number of software faults remaining in each module can be estimated by an SRGM with generalized *logistic* TEF.
2) For each module, the failure data have been collected, and the parameters of each module, including the fault detection rate and the module fault weighting factor, can be estimated.
3) The total amount of testing resource expenditures available for the module testing processes is fixed, and denoted by W.
4) If any of the software modules fails upon execution, the whole software system is in failure.
5) The system manager has to allocate the total testing resources W to each software module, and minimize the number of faults remaining in the system during the testing period. The desired software reliability after the testing phase should achieve the reliability objective R_0.

From Section II-A, the mean value function of a software system with N modules can be formulated as

$$M(t) = \sum_{i=1}^{N} v_i m_i(t) = \sum_{i=1}^{N} v_i a_i\left(1 - \exp\left[-r_i W_i(t)\right]\right). \tag{21}$$

If $v_i = 1$ for all $i = 1, 2, \ldots, N$, the objective is to minimize the total number of faults remaining in the software system after this testing phase. This indicates that the number of remaining faults in the system can be estimated by

$$\sum_{i=1}^{N} v_i a_i \exp\left[-r_i W_i(t)\right] \equiv \sum_{i=1}^{N} v_i a_i \exp[-r_i W_i]. \tag{22}$$

We can further formulate two optimization problems as follows.

A. Minimizing the Number of Remaining Faults With a Given Fixed Amount of TE, and a Reliability Objective

A successful test is one that uncovers an as-yet-undiscovered fault. We should know that tests show the presence, not the absence, of defects [3]. It is impossible to execute every combination of paths during testing. The Pareto principle implies that 80 percent of all faults uncovered during testing will likely be traceable to 20 percent of all program components [3]. Thus the question of how much to test is an important economic question. In practice, a fixed amount of TE is generally spent in testing a program. Therefore, the first optimization problem in this paper is that the total amount of TE is fixed, and we want to allocate these efforts to each module in order to minimize the number of remaining faults in the software systems. Suppose the total amount of TE is W, and module i is allocated W_i testing efforts; then the optimization problem can be represented as [11]–[14], [17], [18], [27]

$$\text{Minimize:} \quad \sum_{i=1}^{N} v_i a_i \exp[-r_i W_i] \tag{23}$$

$$\text{Subject to:} \quad \sum_{i=1}^{N} W_i \le W, \quad W_i \ge 0 \tag{24}$$

$$R_i(t) = 1 - \exp[-r_i W_i] \ge R_0 \quad (i = 1, 2, \ldots, N). \tag{25}$$

From (25), we can obtain

$$W_i \ge \frac{-1}{r_i}\ln[1 - R_0], \quad i = 1, 2, \ldots, N. \tag{26}$$

Let $D_i \equiv (-1/r_i)\ln[1 - R_0]$, $i = 1, 2, \ldots, N$. Thus, we have

$$\sum_{i=1}^{N} W_i \le W, \quad W_i \ge 0, \ i = 1, 2, \ldots, N, \text{ and}$$
$$W_i \ge C_i, \text{ where } C_i = \max(0, D_1, D_2, D_3, \ldots, D_N).$$

That is, the optimal testing resource allocation can be specified as below [7]–[9]

$$\text{Minimize:} \quad \sum_{i=1}^{N} v_i a_i \exp[-r_i W_i]$$
$$\text{Subject to:} \quad \sum_{i=1}^{N} W_i \le W, \quad W_i \ge 0 \text{ and } W_i \ge C_i. \tag{27}$$

Let $X_i = W_i - C_i$, then we can transform the above equations to

$$\text{Minimize:} \quad \sum_{i=1}^{N} v_i a_i \exp[-r_i C_i]\exp[-r_i X_i] \tag{28}$$

$$\text{Subject to:} \quad \sum_{i=1}^{N} X_i \le W - \sum_{i=1}^{N} C_i, \quad X_i \ge 0, \quad i = 1, 2, \ldots, N. \tag{29}$$

Note that the parameters v_i, a_i, and r_i should already be estimated by the proposed model. To solve the above problem, the Lagrange multiplier method can be applied. The Lagrange multiplier method transforms the constrained optimization problem into the unconstrained problem by introducing the Lagrange multipliers [22], [27], [31], [32].

Consequently, (28) & (29) can be simplified as

$$\begin{aligned}\text{Minimize:}\quad & L(X_1, X_2, \ldots, X_N, \lambda)\\ &= \sum_{i=1}^{N} v_i a_i \exp[-r_i C_i] \exp[-r_i X_i]\\ &+ \lambda \left(\sum_{i=1}^{N} X_i - W + \sum_{i=1}^{N} C_i \right). \end{aligned} \tag{30}$$

Based on the Kuhn-Tucker conditions (KTC), the necessary conditions for a minimum value of (30) exist, and can be stated as [12]–[15], [31], [32]

$$\mathbf{A1}: \frac{\partial L(X_1, X_2, \ldots, X_N, \lambda)}{\partial X_i} \geq 0, \quad i = 1, 2, \ldots, N. \tag{31a}$$

$$\mathbf{A2}: X_i \frac{\partial L(X_1, X_2, \ldots, X_N, \lambda)}{\partial X_i} = 0, \quad i = 1, 2, \ldots, N. \tag{31b}$$

$$\mathbf{A3}: \lambda \times \left\{ \sum_{i=1}^{N} X_i - \left(W - \sum_{i=1}^{N} C_i \right) \right\} = 0, \quad i = 1, 2, \ldots, N. \tag{31c}$$

Theorem 1: A feasible solution $X_i (i = 1, 2, \ldots, N)$ of (30) is optimal iff

a) $\lambda \geq v_i a_i r_i \exp[-r_i C_i] \times \exp[-r_i X_i]$,
b) $X_i \times \{\lambda - (v_i a_i r_i \exp[-r_i C_i] \times \exp[-r_i X_i])\} = 0$.

Proof:

a) From (30), we have $(\partial L(X_1, X_2, \ldots, X_N, \lambda)/\partial X_i) = -v_i a_i r_i \times \exp[-r_i C_i] \times \exp[-r_i X_i] + \lambda$. Therefore, from (31a), we know that $\lambda \geq v_i a_i r_i \exp[-r_i C_i] \times \exp[-r_i X_i]$.
b) From (30) & (31b), we have $X_i \times \{\lambda - (v_i a_i r_i \exp[-r_i C_i] \times \exp[-r_i X_i])\} = 0$.

Corollary 1: Let X_i be a feasible solution of (13)

a) $X_i = 0$ iff $\lambda \geq v_i a_i r_i \exp[-r_i C_i]$.
b) If $X_i > 0$, then $X_i = \{\ln(v_i a_i r_i \exp[-r_i C_i]) - \ln \lambda\}/r_i$.

Proof:

a) If $X_i = 0$, then Theorem 1 part a) implies that $\lambda \geq v_i a_i r_i \exp[-r_i C_i]$. Besides, if $\lambda = v_i a_i r_i \exp[-r_i C_i]$, then from Theorem 1 part b), we know that $X_i \times \{v_i a_i r_i \exp[-r_i C_i] - v_i a_i r_i \exp[-r_i C_i] \times \exp[-r_i X_i]\} = 0$ or $X_i \times v_i a_i r_i \exp[-r_i C_i] \times \{1 - \exp[-r_i X_i]\} = 0$. Because $v_i \neq 0$, $a_i \neq 0$, and $r_i \neq 0$, we have $X_i = 0$ or $1 - \exp[-r_i X_i] = 0$, i.e., $X_i = 0$. That is, $X_i = 0$. If $\lambda > v_i a_i r_i \exp[-r_i C_i]$, then $\lambda > v_i a_i r_i \exp[-r_i C_i] \geq v_i a_i r_i \exp[-r_i C_i] \times \exp[-r_i X_i]$ (because $\exp[-r_i X_i] \leq 1$) or $\lambda - v_i a_i r_i \exp[-r_i C_i] \times \exp[-r_i X_i] \neq 0$. Therefore, from Theorem 1 part b), we have $X_i = 0$. Q.E.D.
b) From Theorem 1 part b), we know that if $X_i > 0$, then $\lambda - v_i a_i r_i \exp[-r_i C_i] \times \exp[-r_i X_i] = 0$. Therefore, $X_i = \{\ln(v_i a_i r_i \exp[-r_i C_i]) - \ln \lambda\}/r_i$. Q.E.D.

From (30), we have

$$\begin{aligned}\frac{\partial L(X_1, X_2, \ldots, X_N, \lambda)}{\partial X_i} &= -v_i a_i r_i \times \exp[-r_i C_i]\\ &\quad \times \exp[-r_i X_i] + \lambda = 0\\ \frac{\partial L(X_1, X_2, \ldots, X_N, \lambda)}{\partial \lambda} &= \sum_{i=1}^{N} X_i - W + \sum_{i=1}^{N} C_i = 0.\end{aligned}$$

Thus, the solution X_i^0 is

$$X_i^0 = \frac{(\ln(v_i a_i r_i \times \exp[-r_i C_i]) - \ln \lambda^0)}{r_i}, \quad i = 1, 2, \ldots, N. \tag{32}$$

The solution λ° is

$$\lambda^0 = \exp\left[\frac{\sum_{i=1}^{N} \left(\frac{1}{r_i}\right) (\ln v_i a_i r_i \times \exp[-r_i C_i]) - W + \sum_{i=1}^{N} C_i}{\sum_{i=1}^{N} \left(\frac{1}{r_i}\right)}\right]. \tag{33}$$

Hence, we get $X^0 = \left(X_1^0, X_2^0, X_3^0, \ldots, X_N^0\right)$ as an optimal solution to (30). However, the above X^0 may have some negative components if $v_i a_i r_i \times \exp[-r_i C_i] < \lambda^0$, making X^0 infeasible for (28) & (29). If this is the case, the solution X^0 can be corrected by the following steps [4], [5], [10]:

Algorithm 1

Step 1: Set $l = 0$.

Step 2: Calculate the equations

$$X_i = \frac{1}{r_i}\left[\ln\left(v_i a_i r_i \times \exp[-r_i C_i]\right) - \ln \lambda\right], i = 1, 2, \ldots, N - l.$$

$$\lambda = \exp\left[\frac{\sum_{i=1}^{N-l} \left(\frac{1}{r_i}\right)\left(\ln v_i a_i r_i \times \exp\left[-r_i C_i\right]\right) - W + \sum_{i=1}^{N} C_i}{\sum_{i=1}^{N} \left(\frac{1}{r_i}\right)}\right].$$

Step 3: Rearrange the index i such that $\left(X_1^* \geq X_2^* \geq \cdots \geq X_{N-l}^*\right)$.

Step 4: If $X_{N-l}^* \geq 0$, then stop (i.e., the solution is optimal)

Else, $X_{N-l}^* = 0$; $l = l + 1$.

End If.

Step 5: Go to **Step 2**.

The optimal solution has the form shown in (34) at the bottom of the next page. Algorithm 1 always converges in, at worst,

$N-1$ steps. Thus, the value of the objective function given by (28) at the optimal solution $(X_1^*, X_2^*, \ldots, X_N^*)$ as

$$\sum_{i=1}^{N} v_i a_i \exp[-r_i C_i] \exp\left[-r_i X_i^*\right]. \tag{35}$$

B. Minimizing the Amount of TE Given the Number of Remaining Faults, and a Reliability Objective

Now suppose Z specifies the number of remaining faults in the system, and we have to allocate an amount of TE to each software module to minimize the total TE. The optimization problem can then be represented as

$$\text{Minimize:} \quad \sum_{i=1}^{N} W_i, \tag{36}$$

$$\text{Subject to:} \quad \sum_{i=1}^{N} v_i a_i \exp[-r_i W_i] \le Z, \quad W_i \ge 0 \tag{37}$$

$$R_i(t) = 1 - \exp[-r_i W_i] \ge R_0. \tag{38}$$

Similarly, from (38), we can obtain W_i

$$W_i \ge \frac{-1}{r_i} \ln(1 - R_0), \quad i = 1, 2, \ldots, N. \tag{39}$$

Following similar steps described in Section III-A and letting $X_i = W_i - C_i$, where $C_i = \max(0, D_1, D_2, D_3, \ldots, D_N)$, we can transform the above equations to

$$\text{Minimize:} \quad \sum_{i=1}^{N} (X_i + C_i), \tag{40}$$

$$\text{Subject to:} \quad \sum_{i=1}^{N} v_i a_i \exp[-r_i C_i] \exp[-r_i X_i] \le Z, \quad X_i + C_i \ge 0. \tag{41}$$

To solve this problem, the Lagrange multiplier method can again be used. Equation (40) & (41) are combined to the equation

$$\text{Minimize:} \quad L(X_1, X_2, \ldots, X_N, \lambda) = \sum_{i=1}^{N} (X_i + C_i) + \lambda \left(\sum_{i=1}^{N} v_i a_i \exp[-r_i C_i] \exp[-r_i X_i] - Z \right). \tag{42}$$

Theorem 2: A feasible solution $X_i (i = 1, 2, \ldots, N)$ of (42) is optimal iff

a) $\lambda \le \exp[r_i(C_i + X_i)]/v_i a_i r_i$,
b) $X_i \times \{1 - \lambda v_i a_i r_i \exp[-r_i C_i] \exp[-r_i X_i]\} = 0$

Proof:

a) From (42), we know that $(\partial L(X_1, X_2, \ldots, X_N, \lambda)/\partial X_i) = 1 - \lambda v_i a_i r_i \exp[-r_i C_i] \exp[-r_i X_i]$. Besides, from (31a), we have $1 - \lambda v_i a_i r_i \exp[-r_i C_i] \exp[-r_i X_i] \ge 0$, i.e., $\lambda v_i a_i r_i \exp[-r_i C_i] \exp[-r_i X_i] \le 1$. Therefore, $\lambda \le \exp[r_i(C_i + X_i)]/v_i a_i r_i$.
b) From (42) & (31b), we have $X_i \times \{1 - \lambda v_i a_i r_i \exp[-r_i C_i] \exp[-r_i X_i]\} = 0$.

Corollary 2: Let X_i be a feasible solution of (42)

a) $X_i = 0$ iff $\lambda \le \exp[r_i(C_i)]/v_i a_i r_i$.
b) If $X_i > 0$, then $X_i = \ln(\lambda v_i a_i r_i \exp[-r_i C_i])/r_i$.

Proof:

a) If $X_i = 0$, then Theorem 2 part a) implies that $\lambda \le \exp[r_i(C_i)]/v_i a_i r_i$. Besides, if $\lambda = \exp[r_i(C_i)]/v_i a_i r_i$, then from Theorem 2 part b), we know that $X_i \times \{1 - \exp[-r_i X_i]\} = 0$. Thus, we have $X_i = 0$ or $1 - \exp[-r_i X_i] = 0$, i.e., $X_i = 0$. That is, $X_i = 0$. If $\lambda < \exp[r_i(C_i)]/v_i a_i r_i$, that is, $\lambda v_i a_i r_i < \exp[r_i(C_i)]$ or $\lambda v_i a_i r_i \exp[-r_i(C_i)] < 1$. Hence, $\lambda v_i a_i r_i \exp[-r_i C_i] \exp[-r_i X_i] < \exp[-r_i X_i]$. Because $\exp[-r_i X_i] \le 1$, then we have $\lambda v_i a_i r_i \exp[-r_i C_i] \exp[-r_i X_i] \le 1$ or $1 - \lambda v_i a_i r_i \exp[-r_i C_i] \exp[-r_i X_i] \ne 0$. Therefore, from Theorem 2 part b), we have $X_i = 0$. Q.E.D.
b) From Theorem 2 part b), we know that if $X_i > 0$, $\lambda v_i a_i r_i \exp[-r_i C_i] \exp[-r_i X_i] = 1$. Therefore, $X_i = \ln(\lambda v_i a_i r_i \exp[-r_i C_i])/r_i$. Q.E.D.

From (42), we have

$$\frac{\partial L(X_1, X_2, \ldots, X_N, \lambda)}{\partial X_i} = -\lambda v_i a_i r_i \exp[-r_i C_i] \exp[-r_i X_i] + 1 = 0 \tag{43}$$

$$\frac{\partial L(X_1, X_2, \ldots, X_N, \lambda)}{\partial \lambda} = \sum_{i=1}^{N} v_i a_i \exp[-r_i C_i] \exp[-r_i X_i] - Z = 0. \tag{44}$$

Thus, the solution X_i^0 is

$$X_i^0 = \frac{\ln\left(\lambda v_i a_i r_i \exp[-r_i C_i]\right)}{r_i}, \quad i = 1, 2, \ldots, N. \tag{45}$$

$$\begin{cases} X_i^* = \dfrac{\left(\ln(v_i a_i r_i) \times \exp[-r_i C_i] - \ln \lambda\right)}{r_i}, & i = 1, 2, \ldots, N-l, \text{ where } \lambda = \exp\left[\dfrac{\sum_{i=1}^{N-l} \left(\frac{1}{r_i}\right)\left(\ln v_i a_i r_i \times \exp[-r_i C_i]\right) - W + \sum_{i=1}^{N} C_i}{\sum_{i=1}^{N-l} \left(\frac{1}{r_i}\right)}\right] \\ X_i^* = 0, & \text{otherwise.} \end{cases} \tag{34}$$

TABLE I
THE ESTIMATED VALUES OF a_i, r_i, v_i, AND κ

Module	a_i	r_i	κ	v_i in Example1	v_i in Example 2	v_i in Example 3
1	89	4.18×10^{-4}	1	1.0	1.0	0.5
2	25	5.09×10^{-4}	1	1.5	0.6	0.5
3	27	3.96×10^{-4}	1	1.3	0.7	0.7
4	45	2.30×10^{-4}	1	0.5	0.4	0.4
5	39	2.53×10^{-4}	1	2.0	1.0	1.5
6	39	1.72×10^{-4}	1	0.3	0.2	0.2
7	59	8.82×10^{-5}	1	1.7	0.5	0.6
8	68	7.27×10^{-5}	1	1.3	0.6	0.6
9	37	6.82×10^{-5}	1	1.0	0.1	0.9
10	14	1.53×10^{-4}	1	1.0	0.5	0.5

The solution λ° is

$$\lambda^0 = \frac{\sum_{i=1}^{N}\left(\frac{1}{r_i}\right)}{Z}. \tag{46}$$

That is

$$X_i^0 = \frac{\left(\ln\left(\frac{v_i a_i r_i}{Z}\exp[-r_i C_i]\sum_{i=1}^{N}\frac{1}{r_i}\right)\right)}{r_i}, \quad i = 1, 2 \ldots, N. \tag{47}$$

Hence, we get $X^0 = \left(X_1^0, X_2^0, X_3^0, \ldots, X_N^0\right)$ as an optimal solution to (42). However, the above X^0 may have some negative components if $v_i a_i r_i \exp[-r_i C_i] < \left(Z/\sum_{i=1}^{N}(1/r_i)\right)$, making X^0 infeasible for (40) & (41). In this case, the solution X^0 can be corrected by the following steps. Similarly, we propose a simple algorithm to determine the optimal solution for the TE allocation problem.

Algorithm 2
Step 1: Set $l = 0$.
Step 2: Calculate

$$X_i^* = \frac{1}{r_i}\left[\ln\left(\frac{v_i a_i r_i}{Z}\exp[-r_i C_i]\sum_{i=1}^{N-l}\frac{1}{r_i}\right)\right], \quad i = 1, 2, \ldots, N-l.$$

Step 3: Rearrange the index i such that $X_1^* \geq X_2^* \geq \cdots \geq X_{N-l}^*$.
Step 4: If $X_{N-l}^* \geq 0$ then stop.
Else update $X_{N-l}^* = 0$; $l = l + 1$.
End If.
Step 5: Go to **Step 2**.

The optimal solution has the form

$$X_i^* = \frac{1}{r_i}\left[\ln\left(\frac{v_i a_i r_i}{Z}\exp[-r_i C_i]\sum_{i=1}^{N-l}\frac{1}{r_i}\right)\right], \quad i = 1, 2, \ldots, N-l. \tag{48}$$

Algorithm 2 always converges in, at worst, $N - 1$ steps.

TABLE II
THE OPTIMAL TE EXPENDITURES USING ALGORITHM 1

Module	X_i^* for Example 1	X_i^* for Example 2	X_i^* for Example 3
1	6254	8105	6015
2	3826	3547	2833
3	4117	4409	4052
4	2791	5191	4402
5	7825	8145	9030
6	0	403	0
7	13366	8267	8280
8	11820	11833	9343
9	0	0	6046
10	0	0	0

IV. EXPERIMENTAL STUDIES AND RESULTS

In this section, three cases for the optimal TE allocation problems are demonstrated. Here we assume that the estimated parameters a_i & r_i in (21), for a software system consisting of 10 modules, are summarized in Table I. Moreover, the weighting vectors v_i in (21) are also listed. In the following, we illustrate several examples to show how the optimal allocation of TE expenditures to each software module is determined. Suppose that the total amount of TE expenditures W is 50 000 man-hours, and $R_0 = 0.9$. Besides, all the parameters a_i & r_i of (21) for each software module have been estimated by using the method of MLE or LSE in Section II-B. We apply the proposed model to software failure data set [12], [15], [27], [33]. Here we have to allocate the expenditures to each module, and minimize the number of remaining faults. From Table I & Algorithm 1 in Section III-A, the optimal TE expenditures for the software systems are estimated, and shown in Table II.

For example, using the estimated parameters a_i, r_i, the weighting factor v_i in Table I, and the optimal TE expenditures in Table II, the value of the estimated number of remaining faults is 172 for Example 1. That is, the total number of remaining faults is intended to decrease from 514 to 172 by using testing-resource expenditures of 50 000 man-hours, and about a 33.6% reduction in the number of remaining faults. Conversely, if we want to decrease more remaining faults, and get a better reduction rate, then we have to re-plan & consider the allocation of testing-resource expenditures; i.e., using the same values of a_i, r_i, κ, and v_i, the optimal TE expenditures should be re-estimated. Therefore, we can know how much

TABLE III
THE REDUCTION IN THE NUMBER OF REMAINING FAULTS

Example	Initial faults	Remaining faults	Reduction (%)
1	514.0	172.0	33.6
2	268.7	68.5	25.5
3	276.7	97.4	35.2

TABLE IV
THE OPTIMAL TE EXPENDITURES USING ALGORITHM 2

Module	X_i^* for Example 1	X_i^* for Example 2	X_i^* for Example 3
1	7700	6962	5941
2	4976	2608	2772
3	5692	3302	3974
4	5669	3109	4268
5	10168	6258	8908
6	2096	0	0
7	20505	2847	7931
8	20293	5263	8919
9	7265	0	5595
10	2388	0	0

extra amount of testing-resource expenditures is expected [12], [15]. The numbers of initial faults, the estimated remaining faults, and the reduction in the number of remaining faults for the other examples are shown in Table III.

Finally, suppose the total number of remaining faults Z is 100. We have to allocate the expenditures to each module, and minimize the total amount of TE expenditures. Similarly, using Algorithm 2 in Section III-B & Table I, the optimal solutions of TE expenditures are derived & shown in Table IV. Furthermore, the relationship between the total amount of testing-effort expenditures, and the reduction rate of the remaining faults, are also depicted in Fig. 1.

V. SENSITIVITY ANALYSIS

In this section, sensitivity analysis of the proposed model is conducted to study the effect of the principal parameters, such as the *expected initial faults*, and the *fault detection rate*. In (4), we know that there are some parameters affecting the mean value function, such as the expected total number of initial faults, the fault detection rate, the total amount of TE, the consumption rate of TE expenditures, and the structuring index, etc. Consequently, we have to estimate all these parameters for each software module very carefully because they play an important role for the optimal resource allocation problems. In general, each parameter is estimated based on the available data, which is often sparse. Thus, we analyze the sensitivity of some principal parameters, but not all parameters due to the limitation of space. Nevertheless, we still can evaluate the optimal resource allocation problems for various conditions by examining the behavior of some parameters with the most significant influence. We perform the sensitivity analysis of optimal resource allocation problems with respect to the estimated parameters so that attention can be paid to those parameters deemed critical [34]–[40]. In this paper, we define

$$\text{Relative Change } (RC) = \frac{MTEE - OTEE}{OTEE} \tag{49}$$

where *OTEE* is the original optimal TE expenditures, and *MTEE* is the modified optimal TE expenditures.

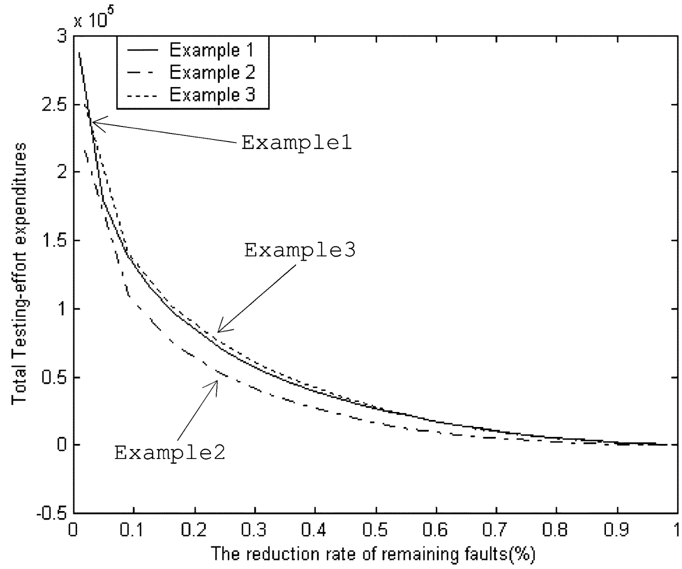

Fig. 1. The reduction rate of remaining faults vs. the total TE expenditures.

TABLE V
SOME NUMERICAL VALUES OF THE OPTIMAL TE EXPENDITURES FOR THE CASES OF 40%, 30%, 20%, AND 10% INCREASE IN a_1 (ALGORITHM 1)

Module	X_i^* $(a_1 \times 1.4)$	X_i^* $(a_1 \times 1.3)$	X_i^* $(a_1 \times 1.2)$	X_i^* $(a_1 \times 1.1)$
1	7011	6844	6664	6469
2	3787	3795	3805	3815
3	4067	4078	4090	4103
4	2704	2723	2744	2766
5	7746	7764	7782	7803
6	0	0	0	0
7	13139	13189	13243	13301
8	11546	11606	11672	11743
9	0	0	0	0
10	0	0	0	0

A. Effect of Variations on Expected Initial Faults & Fault Detection Rate (Algorithm 1)

Assuming we have obtained the optimal TE expenditures to each software module that minimize the expected cost of software, then we can calculate the *MTEE* concerning the changes of expected number of initial faults a_i for the specific module i. The procedure can be repeated for various values of a_i. For instance, for the data set used in Section IV (here we only use Example 1 as illustration), if the expected number of initial faults a_1 of module 1 is increased or decreased by 40%, 30%, 20%, or 10%, then the modified TE expenditures for each software module can be obtained by following the similar procedures. Table V shows some numerical values of the optimal TE expenditures for the case of 40%, 30%, 20%, and 10% increase to a_1. The result indicates that the estimated values of optimal TE expenditures will be changed when a_1 changes. That is, if a_1 is increased by 40%, then the estimated value of optimal TE expenditure for module 1 is changed from 6254 to 7011, and its RC is 0.121 (about 12% increment). But for modules 2, 3, 4, 5, 7, and 8, the estimated values of optimal TE expenditures are about 1.02%, 1.21%, 0.12%, 1.01%, 1.69%, and 2.32% decrement, respectively. Therefore, the variation in a_1 has the most significant influence on the optimal allocation of TE expenditures. From Table V, we can also know that, if the change of a_1

TABLE VI
SOME NUMERICAL VALUES OF THE OPTIMAL TE EXPENDITURES FOR THE CASES OF 40%, 30%, 20%, AND 10% DECREASE IN a_1 (ALGORITHM 1)

Module	X_i^* (a_1×0.6)	X_i^* (a_1×0.7)	X_i^* (a_1×0.8)	X_i^* (a_1×0.9)
1	5105	5452	5752	6017
2	3886	3868	3852	3838
3	4194	4171	4151	4133
4	2923	2883	2849	2818
5	7945	7909	7877	7850
6	0	0	0	0
7	13710	13606	13516	13437
8	12237	12112	12003	11906
9	0	0	0	0
10	0	0	0	0

TABLE VII
SOME NUMERICAL VALUES OF THE OPTIMAL TE EXPENDITURES FOR THE CASES OF 40%, 30%, 20%, AND 10% INCREASE IN a_1 & a_2 (ALGORITHM 1)

Module	X_i^* (a_1×1.4 & a_2×1.4)	X_i^* (a_1×1.3 & a_2×1.3)	X_i^* (a_1×1.2 & a_2×1.2)	X_i^* (a_1×1.1 & a_2×1.1)
1	6972	6814	6643	6447
2	4415	4285	4145	4155
3	4026	4046	4068	4081
4	2633	2668	2705	2728
5	7682	7713	7747	7768
6	0	0	0	0
7	12953	13044	13142	13201
8	11320	11430	11549	11620
9	0	0	0	0
10	0	0	0	0

is small, the sensitivity of the optimal testing-resources allocation with respect to the value of a_1 is low. Next, from Table VI, it is shown that, if a_1 is decreased by 30%, the estimated value of optimal TE expenditure for module 1 is changed from 6254 to 5452, and its RC is -0.128 (about 12.8% decrement). It is noted that for modules 2, 3, 4, 5, 7, and 8, the estimated values of optimal TE expenditures are about 1.10%, 1.31%, 3.29%, 1.07%, 1.79%, and 2.47% increment, respectively.

We have performed an extensive sensitivity analysis for the expected initial faults as shown above. But each a_i is considered in isolation. Now we try to study the effects of simultaneous changes of a_i & $a_j (j \neq i)$. If we let a_1 & a_2 both be increased by 40%, then the estimated values of optimal TE expenditure for modules 1, and 2 are changed from 6254 to 6972 (about 11.48% increment), and 3826 to 4415 (about 15.39% increment), respectively. But for modules 3, 4, 5, 7, and 8, the estimated values of optimal TE expenditures are about 2.21%, 5.66%, 1.83%, 3.09%, and 4.23% decrement, respectively. Therefore, the variation in a_1 & a_2 has the significant influence on the optimal allocation of TE expenditures. Similarly, from Table VII, we can also know that if the changes of a_1 & a_2 are less, the sensitivity of the optimal testing-resources allocation with respect to the values of a_1 & a_2 is low. From Table VIII, it is also shown that if a_1 & a_2 are both decreased by 30%, the estimated values of optimal TE expenditure for modules 1, and 2 are changed from 6254 to 5494 (about 12.2% decrement), and 3826 to 3201 (about 16.3% decrement), respectively. It is also noted that for module 3, 4, 5, 7, and 8, the estimated values of optimal TE expenditures are, respectively, about 2.38%, 6.02%, 1.94%, 3.27%, and 4.49% increment. Based on these observations, we can conclude that if a_i is changed, it will have much influence on the estimated values of optimal TE expenditure for module i.

TABLE VIII
SOME NUMERICAL VALUES OF THE OPTIMAL TE EXPENDITURES FOR THE CASES OF 40%, 30%, 20%, AND 10% DECREASE IN a_1 & a_2 (ALGORITHM 1)

Module	X_i^* (a_1×0.6 & a_2×0.6)	X_i^* (a_1×0.7 & a_2×0.7)	X_i^* (a_1×0.8 & a_2×0.8)	X_i^* (a_1×0.9 & a_2×0.9)
1	5165	5494	5778	6030
2	2931	3201	3435	3641
3	4257	4215	4178	4146
4	3032	2959	2896	2841
5	8043	7977	7920	7870
6	0	0	0	0
7	13992	13803	13639	13495
8	12580	12351	12152	11977
9	0	0	0	0
10	0	0	0	0

TABLE IX
SOME NUMERICAL VALUES OF THE OPTIMAL TE EXPENDITURES FOR THE CASES OF 40%, 30%, 20%, AND 10% INCREASE IN r_1 (ALGORITHM 1)

Module	X_i^* (r_1×1.4)	X_i^* (r_1×1.3)	X_i^* (r_1×1.2)	X_i^* (r_1×1.1)
1	5094	5338	5609	5912
2	3886	3873	3859	3844
3	4195	4178	4160	4140
4	2925	2897	2865	2830
5	7946	7921	7892	7861
6	0	0	0	0
7	13713	13640	13559	13468
8	12241	12153	12055	11944
9	0	0	0	0
10	0	0	0	0

TABLE X
SOME NUMERICAL VALUES OF THE OPTIMAL TE EXPENDITURES FOR THE CASES OF 40%, 30%, 20%, AND 10% DECREASE IN r_1 (ALGORITHM 1)

Module	X_i^* (r_1×0.6)	X_i^* (r_1×0.7)	X_i^* (r_1×0.8)	X_i^* (r_1×0.9)
1	8185	7595	7085	6642
2	3726	3756	3783	3806
3	3989	4028	4062	4092
4	2569	2637	2696	2747
5	7624	7685	7739	7785
6	0	0	0	0
7	12788	12964	13117	13250
8	11120	11334	11519	11680
9	0	0	0	0
10	0	0	0	0

TABLE XI
SOME NUMERICAL VALUES OF THE OPTIMAL TE EXPENDITURES FOR THE CASES OF 40%, 30%, 20%, AND 10% INCREASE IN r_1 & r_2 (ALGORITHM 1)

Module	X_i^* (r_1×1.4 & r_2×1.4)	X_i^* (r_1×1.3 & r_2×1.3)	X_i^* (r_1×1.2 & r_2×1.2)	X_i^* (r_1×1.1 & r_2×1.1)
1	5122	5361	5626	5922
2	3271	3395	3529	3672
3	4236	4210	4182	4152
4	2996	2952	2903	2850
5	8011	7971	7927	7878
6	0	0	0	0
7	13898	13784	13658	13519
8	12466	12327	12174	12006
9	0	0	0	0
10	0	0	0	0

In fact, we can investigate the sensitivity of fault detection rate following the similar steps described above. Table IX shows numerical values of the optimal TE expenditures for the case of 40%, 30%, 20%, and 10% increase to r_1. Table X shows numerical values of the optimal TE expenditures for the cases of 40%, 30%, 20%, and 10% decrease in r_1. Numerical values of the optimal TE expenditures for the cases of 40%, 30%, 20%, and 10% increase in r_1 & r_2 are shown in Table XI. Finally,

TABLE XII
SOME NUMERICAL VALUES OF THE OPTIMAL TE EXPENDITURES FOR THE CASES OF 40%, 30%, 20%, AND 10% DECREASE IN r_1 & r_2 (ALGORITHM 1)

Module	X_i^* ($r_1\times0.6$ & $r_2\times0.6$)	X_i^* ($r_1\times0.7$ & $r_2\times0.7$)	X_i^* ($r_1\times0.8$ & $r_2\times0.8$)	X_i^* ($r_1\times0.9$ & $r_2\times0.9$)
1	8110	7546	7057	6629
2	4476	4325	4157	3989
3	3941	3992	4038	4080
4	2487	2574	2654	2726
5	7550	7628	7701	7766
6	0	0	0	0
7	12574	12800	13008	13196
8	10861	11135	11387	11615
9	0	0	0	0
10	0	0	0	0

TABLE XIII
SOME NUMERICAL VALUES OF THE OPTIMAL TE EXPENDITURES FOR THE CASES OF 40%, 30%, 20%, AND 10% INCREASE IN a_1 (ALGORITHM 2)

Module	X_i^* ($a_1\times1.4$)	X_i^* ($a_1\times1.3$)	X_i^* ($a_1\times1.2$)	X_i^* ($a_1\times1.1$)
1	8504	8327	8136	7928
2	5013	5013	5013	5013
3	5643	5643	5643	5643
4	5424	5424	5424	5424
5	10211	10211	10211	10211
6	1770	1770	1770	1770
7	20220	20220	20220	20220
8	20131	20131	20131	20131
9	7759	7759	7759	7759
10	2388	2388	2388	2388

Table XII shows numerical values of the optimal TE expenditures for the cases of 40%, 30%, 20%, and 10% decrease in r_1 & r_2.

B. *Effect of Variations on Expected Initial Faults & Fault Detection Rate (Algorithm 2)*

Assuming we have obtained the optimal TE expenditures to each software module, then we can calculate the *MTEE* concerning the changes of expected number of initial faults a_i for the specific module i. The procedure can be repeated for various values of a_i. Similarly, we investigate the possible change of optimal TE expenditures when the expected number of initial faults a_1 is changed. For the data set (Example 1) used in Section IV, if the expected number of initial faults a_1 of module 1 is increased or decreased by 40%, 30%, 20%, or 10%, then the modified TE expenditures for each software module can be re-estimated from the algorithms in Section III. First, Table XIII shows some numerical values of the optimal TE expenditures for the cases of 40%, 30%, 20%, and 10% increase in a_1. The result indicates that the estimated values of optimal TE expenditures will be changed when a_1 changes.

For example, if a_1 is increased by 40%, then the estimated value of optimal TE expenditure for module 1 is changed from 7700 to 8504, and its RC is 0.104 (about 10% increment). Besides, for modules 3, 4, 6, 7, and 8, the estimated values of optimal TE expenditures are about 0.86%, 4.32%, 15.55%, 1.39%, and 0.798% decrement, respectively. But for modules 2, 5, and 9, the estimated values of optimal TE expenditures are about 0.74%, 0.42%, and 6.79% increment, respectively. Therefore, from Table XIII, we can know that, if the change of a_1 is small, the sensitivity of the optimal testing-resources allocation with respect to the value of a_1 is low. Next, we show the same comparison results in case that a_1 is decreased. From Table XIV, it is shown that, if a_1 is decreased by 30%, the estimated value of optimal TE expenditure for module 1 is changed from 7700 to 6847, and its RC is -0.111 (about 10.7% decrement).

TABLE XIV
SOME NUMERICAL VALUES OF THE OPTIMAL TE EXPENDITURES FOR THE CASES OF 40%, 30%, 20%, AND 10% DECREASE IN a_1 (ALGORITHM 2)

Module	X_i^* ($a_1\times0.6$)	X_i^* ($a_1\times0.7$)	X_i^* ($a_1\times0.8$)	X_i^* ($a_1\times0.9$)
1	6478	6847	7166	7448
2	5013	5013	5013	5013
3	5643	5643	5643	5643
4	5424	5424	5424	5424
5	10211	10211	10211	10211
6	1770	1770	1770	1770
7	20220	20220	20220	20220
8	20131	20131	20131	20131
9	7759	7759	7759	7759
10	2388	2388	2388	2388

TABLE XV
SOME NUMERICAL VALUES OF THE OPTIMAL TE EXPENDITURES FOR THE CASES OF 40%, 30%, 20%, AND 10% INCREASE IN a_1 &a_2 (ALGORITHM 2)

Module	X_i^* ($a_1\times1.4$ & $a_2\times1.4$)	X_i^* ($a_1\times1.3$ & $a_2\times1.3$)	X_i^* ($a_1\times1.2$ & $a_2\times1.2$)	X_i^* ($a_1\times1.1$ & $a_2\times1.1$)
1	8504	8327	8136	7928
2	5674	5528	5371	5371
3	5643	5643	5643	5643
4	5424	5424	5424	5424
5	10211	10211	10211	10211
6	1770	1770	1770	1770
7	20220	20220	20220	20220
8	20131	20131	20131	20131
9	7759	7759	7759	7759
10	2388	2388	2388	2388

TABLE XVI
SOME NUMERICAL VALUES OF THE OPTIMAL TE EXPENDITURES FOR THE CASES OF 40%, 30%, 20%, AND 10% DECREASE IN a_1 & a_2 (ALGORITHM 2)

Module	X_i^* ($a_1\times0.6$ & $a_2\times0.6$)	X_i^* ($a_1\times0.7$ & $a_2\times0.7$)	X_i^* ($a_1\times0.8$ & $a_2\times0.8$)	X_i^* ($a_1\times0.9$ & $a_2\times0.9$)
1	6478	6847	7166	7448
2	4010	4313	4575	4806
3	5643	5643	5643	5643
4	5424	5424	5424	5424
5	10211	10211	10211	10211
6	1770	1770	1770	1770
7	20220	20220	20220	20220
8	20131	20131	20131	20131
9	7759	7759	7759	7759
10	2388	2388	2388	2388

So far, we have performed an extensive sensitivity analysis for the expected initial faults as shown above. However, each a_i is considered in isolation. Again we study the effects of simultaneous changes of a_i & $a_j (j \neq i)$. If we let a_1 & a_2 both be increased by 40%, then the estimated values of optimal TE expenditure for modules 1, and 2 are changed from 7700 to 8504 (about 10.4% increment), and 4976 to 5674 (about 14.0% increment), respectively. From Table XV, we can find that the variation in a_1 & a_2 may have the most significant influence on the optimal allocation of TE expenditures.

Similarly, we can also know that, if the changes of a_1 & a_2 are less, the sensitivity of the optimal testing-resources allocation with respect to the values of a_1 & a_2 is low. From Table XVI, we can see that, if a_1 & a_2 are both decreased by 30%, the estimated values of optimal TE expenditure for modules 1, and 2 are changed from 7700 to 6847 (about 11.1% decrement), and

TABLE XVII
SOME NUMERICAL VALUES OF THE OPTIMAL TE EXPENDITURES FOR THE CASES OF 40%, 30%, 20%, AND 10% INCREASE IN r_1 (ALGORITHM 2)

Module	X_i^* (r_1×1.4)	X_i^* (r_1×1.3)	X_i^* (r_1×1.2)	X_i^* (r_1×1.1)
1	6507	6390	6768	7200
2	4993	4997	5001	5007
3	5618	5623	5628	5635
4	5380	5389	5399	5410
5	10171	10179	10188	10198
6	1711	1723	1736	1752
7	20104	20127	20153	20183
8	19990	20018	20049	20086
9	7610	7639	7672	7712
10	2322	2335	2350	2367

TABLE XVIII
SOME NUMERICAL VALUES OF THE OPTIMAL TE EXPENDITURES FOR THE CASES OF 40%, 30%, 20%, AND 10% DECREASE IN r_1 (ALGORITHM 2)

Module	X_i^* (r_1×0.6)	X_i^* (r_1×0.7)	X_i^* (r_1×0.8)	X_i^* (r_1×0.9)
1	10890	9833	8984	8286
2	5059	5043	5030	5021
3	5703	5682	5666	5653
4	5526	5490	5463	5442
5	10304	10271	10246	10227
6	1906	1858	1822	1793
7	20486	20392	20320	20265
8	20453	20339	20252	20185
9	8103	7981	7889	7817
10	2541	2487	2446	2414

TABLE XIX
SOME NUMERICAL VALUES OF THE OPTIMAL TE EXPENDITURES FOR THE CASES OF 40%, 30%, 20%, AND 10% INCREASE IN r_1 & r_2 (ALGORITHM 2)

Module	X_i^* (r_1×1.4 & r_2×1.4)	X_i^* (r_1×1.3 & r_2×1.3)	X_i^* (r_1×1.2 & r_2×1.2)	X_i^* (r_1×1.1 & r_2×1.1)
1	6042	6378	6758	7194
2	4027	4230	4458	4717
3	5596	5605	5616	5629
4	5343	5359	5377	5399
5	10137	10152	10168	10188
6	1662	1683	1707	1736
7	20008	20049	20097	20153
8	19874	19924	19982	20050
9	7486	7539	7600	7673
10	2266	2290	2317	2350

TABLE XX
SOME NUMERICAL VALUES OF THE OPTIMAL TE EXPENDITURES FOR THE CASES OF 40%, 30%, 20%, AND 10% DECREASE IN r_1 & r_2 (ALGORITHM 2)

Module	X_i^* (r_1×0.6 & r_2×0.6)	X_i^* (r_1×0.7 & r_2×0.7)	X_i^* (r_1×0.8 & r_2×0.8)	X_i^* (r_1×0.9 & r_2×0.9)
1	10965	9875	9006	8294
2	6821	6238	5758	5356
3	5750	5713	5684	5662
4	5609	5544	5494	5456
5	10378	10319	10274	10239
6	2015	1929	1683	1812
7	20699	20530	20402	20301
8	20712	20507	20352	20229
9	8379	8160	7995	7864
10	2664	2567	2493	2435

4976 to 4313 (about 13.3% decrement), respectively. Similarly, we can investigate the sensitivity of fault detection rate for Algorithm 2 following the similar steps described above. Table XVII shows numerical values of the optimal TE expenditures for the cases of 40%, 30%, 20%, and 10% increase in r_1. Table XVIII shows Numerical Values of the Optimal TE Expenditures for the Cases of 40%, 30%, 20%, and 10% Decrease in r_1. Moreover, numerical values of the optimal TE expenditures for the cases of 40%, 30%, 20%, and 10% increase in r_1 & r_2 are shown in Table XIX. Finally, Table XX Shows Numerical Values of the Optimal TE Expenditures for the Cases of 40%, 30%, 20%, and 10% Decrease in r_1 & r_2.

REFERENCES

[1] M. R. Lyu, *Handbook of Software Reliability Engineering*: McGraw Hill, 1996.

[2] J. D. Musa, *Software Reliability Engineering: More Reliable Software, Faster Development and Testing*, New York: McGraw-Hill, 1999.

[3] R. S. Pressman, *Software Engineering: A Practitioner's Approach, 6/e*: McGraw-Hill, 2005.

[4] P. Kubat and H. S. Koch, "Managing test-procedure to achieve reliable software," *IEEE Trans. Rel.*, vol. 32, no. 3, pp. 299–303, 1983.

[5] P. Kubat, "Assessing reliability of modular software," *Operation Research Letters*, vol. 8, no. 1, pp. 35–41, 1989.

[6] B. Littlewood, "Software reliability model for modular program structure," *IEEE Trans. Rel.*, vol. 28, no. 3, pp. 241–4246, 1979.

[7] Y. W. Leung, "Software reliability growth model with debugging efforts," *Microelectronics and Reliability*, vol. 32, no. 5, pp. 699–704, 1992.

[8] ——, "Dynamic resource allocation for software module testing," *The J. Sys. Softw.*, vol. 37, no. 2, pp. 129–139, May 1997.

[9] ——, "Software reliability allocation under uncertain operational profiles," *Journal of the Operational Research Society*, vol. 48, no. 4, pp. 401–411, Apr. 1997.

[10] R. H. Huo, S. Y. Kuo, and Y. P. Chang, "Needed resources for software module test, using the hyper-geometric software reliability growth model," *IEEE Trans. Rel.*, vol. 45, no. 4, pp. 541–549, Dec. 1996.

[11] H. Ohtera and S. Yamada, "Optimal allocation and control problems for software-testing resources," *IEEE Trans. Rel.*, vol. 39, no. 2, pp. 171–176, 1990.

[12] S. Yamada, T. Ichimori, and M. Nishiwaki, "Optimal allocation policies for testing resource based on a software reliability growth model," *International Journal of Mathematical and Computer Modeling*, vol. 22, pp. 295–301, 1995.

[13] M. Nishiwaki, S. Yamada, and T. Ichimori, "Testing-resource allocation policies based on an optimal software release problem," *Mathematica Japonica*, vol. 43, no. 1, pp. 91–97, 1996.

[14] T. Ichimori, H. Masuyama, and S. Yamada, "A two-resource allocation problem according to an exponential objective: Optimum distribution of searching effort," *International Journal of Reliability, Quality and Safety Engineering*, vol. 1, no. 2, pp. 135–146, 1994.

[15] T. Ichimori, "Discrete testing resource allocation in module testing," *International Journal of Reliability, Quality and Safety Engineering*, vol. 6, no. 1, pp. 57–64, 1999.

[16] B. Yang and M. Xie, "Testing-Resource allocation for redundant software systems," in *Proceedings of 1999 Pacific Rim International Symposium on Dependable Computing (PRDC'99)*, Hong Kong, China, Dec. 1999, pp. 78–83.

[17] ——, "Optimal testing-time allocation for modular systems," *International Journal of Quality and Reliability Management*, vol. 18, no. 8, pp. 854–863, 2001.

[18] M. R. Lyu, S. Rangarajan, and A. P. A. van Moorsel, "Optimal allocation of test resources for software reliability growth modeling in software development," *IEEE Trans. Rel.*, vol. 51, no. 2, pp. 183–192, Jun. 2002.

[19] S. Ozekici, K. Altinel, and S. Ozcelikyurek, "Testing of software with an operational profile," *Naval Research Logistics*, vol. 47, pp. 620–634, 2000.

[20] O. Berman and N. Ashrafi, "Optimization models for reliability of modular software systems," *IEEE Trans. Softw. Eng.*, vol. 19, no. 11, pp. 1119–1123, Nov. 1993.

[21] O. Berman and M. Cutler, "Optimal software implementation considering reliability and cost," *Computers and Operations Research*, vol. 25, no. 10, pp. 857–868, 1998.

[22] H. W. Jung and B. Choi, "Optimization models for quality and cost of modular software systems," *European Journal of Operational Research*, pp. 613–619, 1999.

[23] P. K. Kapur, P. C. Jha, and A. K. Bardhan, "Optimal allocation of testing resource for a modular software," *Asia-Pacific Journal of Operational Research*, vol. 21, no. 3, pp. 333–354, 2004.

[24] C. Y. Huang, M. R. Lyu, and S. Y. Kuo, "A unified scheme of some nonhomogenous poisson process models for software reliability estimation," *IEEE Trans. Softw. Eng.*, vol. 29, no. 3, pp. 261–269, Mar. 2003.

[25] S. Y. Kuo, C. Y. Huang, and M. R. Lyu, "Framework for modeling software reliability, using various testing-efforts and fault-detection rates," *IEEE Trans. Rel.*, vol. 50, no. 3, pp. 310–320, Sep. 2001.

[26] C. Y. Huang and S. Y. Kuo, "Analysis and assessment of incorporating logistic testing effort function into software reliability modeling," *IEEE Trans. Rel.*, vol. 51, no. 3, pp. 261–270, Sep. 2002.

[27] C. Y. Huang, J. H. Lo, S. Y. Kuo, and M. R. Lyu, "Optimal allocation of testing resources for modular software systems," in *Proceedings of the IEEE 13th International Symposium on Software Reliability Engineering (ISSRE 2002)*, Annapolis, Maryland, Nov. 2002, pp. 129–138.

[28] ——, "Software reliability modeling and cost estimation incorporating testing-effort and efficiency," in *Proceedings of the IEEE 10th International Symposium on Software Reliability Engineering (ISSRE'99)*, Boca Raton, Florida, Nov. 1999, pp. 62–72.

[29] F. N. Parr, "An alternative to the rayleigh curve for software development effort," *IEEE Trans. Softw. Eng.*, vol. SE-6, pp. 291–296, 1980.

[30] T. DeMarco, *Controlling Software Projects: Management, Measurement and Estimation*: Prentice-Hall, 1982.

[31] G. L. Nemhauser, A. H. G. R. Kan, and M. J. Todd, *Optimization: Handbooks in Operations Research and Management Science*, North-Holland, 1994, vol. 1.

[32] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 2nd ed: John Wiley & Sons, 1993.

[33] J. H. Lo, S. Y. Kuo, M. R. Lyu, and C. Y. Huang, "Optimal resource allocation and reliability analysis for component-based software applications," in *Proceedings of the 26th Annual International Computer Software and Applications Conference (COMPSAC 2002)*, Oxford, England, Aug. 2002, pp. 7–12.

[34] M. Xie and G. Y. Hong, "A study of the sensitivity of software release time," *J. Syst. Softw.*, vol. 44, no. 2, pp. 163–168, 1998.

[35] P. S. F. Yip, X. Liqun, D. Y. T. Fong, and Y. Hayakawa, "Sensitivity-Analysis and estimating number-of-Faults in removal debugging," *IEEE Trans. Rel.*, vol. 48, no. 3, pp. 300–305, 1999.

[36] S. S. Gokhale and K. S. Trivedi, "Reliability prediction and sensitivity analysis based on software architecture," in *Proceedings of the IEEE 13th International Symposium on Software Reliability Engineering (ISSRE 2002)*, Annapolis, Maryland, Nov. 2002, pp. 64–75.

[37] A. Pasquini, A. N. Crespo, and P. Matrella, "Sensitivity of reliability-growth model to operational profile errors vs. Testing accuracy," *IEEE Trans. Rel.*, vol. 45, no. 4, pp. 531–540, 1996.

[38] M. H. Chen, A. P. Mathur, and V. J. Rego, "A case study to investigate sensitivity of reliability estimates to errors in operational profile," in *Proceedings of the IEEE 5th International Symposium on Software Reliability Engineering (ISSRE'94)*, Monterey, California, Oct. 1994, pp. 276–281.

[39] C. Y. Huang, J. H. Lo, J. W. Lin, C. C. Sue, and C. T. Lin, "Optimal resource allocation and sensitivity analysis for software modular testing," in *Proceedings of the IEEE 5th International Symposium on Multimedia Software Engineering (ISMSE 2003)*, Taichung, Taiwan, Dec. 2003, pp. 231–238.

[40] J. H. Lo, C. Y. Huang, S. Y. Kuo, and M. R. Lyu, "Sensitivity analysis of software reliability for distributed component-based software systems," in *Proceedings of the 27th Annual International Computer Software and Applications Conference (COMPSAC 2003)*, Dallas, Texas, Nov. 2003, pp. 500–505.

Chin-Yu Huang (Member IEEE) is currently an Assistant Professor in the Department of Computer Science at National Tsing Hua University, Hsinchu, Taiwan. He received the M.S. (1994), and the Ph.D. (2000) in Electrical Engineering from National Taiwan University, Taipei. He was with the Bank of Taiwan from 1994 to 1999, and was a senior software engineer at Taiwan Semiconductor Manufacturing Company from 1999 to 2000. Before joining NTHU in 2003, he was a division chief of the Central Bank of China, Taipei. His research interests are software reliability engineering, software testing, software metrics, software testability, fault tree analysis, and system safety assessment. He is a member of IEEE.

Michael R. Lyu received the B.S. (1981) in electrical engineering from National Taiwan University; the M.S. (1985) in computer engineering from University of California, Santa Barbara; and the Ph.D. (1988) in computer science from University of California, Los Angeles. He is a Professor in the Computer Science and Engineering Department of the Chinese University of Hong Kong. He worked at the Jet Propulsion Laboratory, Bellcore, and Bell Labs; and taught at the University of Iowa. His research interests include software reliability engineering, software fault tolerance, distributed systems, image & video processing, multimedia technologies, and mobile networks. He has published over 200 papers in these areas. He has participated in more than 30 industrial projects, and helped to develop many commercial systems & software tools. Professor Lyu was frequently invited as a keynote or tutorial speaker to conferences & workshops in U.S., Europe, and Asia. He initiated the International Symposium on Software Reliability Engineering (ISSRE), and was Program Chair for ISSRE'1996, Program Co-Chair for WWW10 & SRDS 2005, and General Chair for ISSRE 2001 & PRDC 2005. He also received Best Paper Awards in ISSRE 98 and in ISSRE 2003. He is the editor-in-chief for two book volumes: Software Fault Tolerance (Wiley, 1995), and the Handbook of Software Reliability Engineering (IEEE and McGraw-Hill, 1996). He has been an Associate Editor of IEEE Transactions on Reliability, IEEE Transactions on Knowledge and Data Engineering, and Journal of Information Science and Engineering. Professor Lyu is an IEEE Fellow.