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.

ACRONYMS1

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

NOTATION

$m(t)$ expected mean number of faults detected in time $(0, t]$, mean value function
$\lambda(t)$ failure intensity for $m(t)$, $dm(t)/dt$
$w_{i}(t)$ current testing-effort consumption at time $t$
$W_{\alpha}(t)$ cumulative testing-effort consumption at time $t$

$\alpha$ expected number of initial faults
$r$ fault detection rate per unit testing-effort
$N$ total amount of testing-effort eventually consumed
$\alpha$ consumption rate of testing-effort expenditures in the generalized logistic testing-effort function
$A$ constant parameter in the generalized logistic testing-effort function
$\kappa$ structuring index whose value is larger for better structured software development efforts
$\beta$ constant parameter
$v_{i}$ weighting factor to measure the relative importance of a fault removal from module $i$
$R(x|t)$ conditional software reliability

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

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.

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

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

\[
\frac{m(t + \Delta t) - m(t)}{w_R(t)} = r \times \left[ a - m(t) \right] \Delta t.
\]

That is

\[
r = \lim_{\Delta t \to 0} \frac{m(t + \Delta t) - m(t)}{a - m(t)} \frac{1}{w_R(t)} \Delta t.
\]

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

\[
\frac{dm(t)}{dt} \times \frac{1}{w_R(t)} = r \times \left[ a - m(t) \right].
\]

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

\[
m(t) = a \left( 1 - \exp \left[ -r(W_R(t) - W_R(0)) \right] \right) = a \left( 1 - \exp \left[ -r(W(t)) \right] \right).
\]
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 Yourend 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

\[ W_k(t) = \frac{N}{\sqrt{1 + Ae^{-\alpha_t}}}. \]  

(5)

The current TE consumption is

\[ w_k(t) = \frac{du_k(t)}{dt}. \]  

(6)

The TE reaches its maximum value at time

\[ t_{max} = \ln \frac{4}{\kappa \alpha}. \]  

(7)

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

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

(8)

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

\[ \ln R_{cond}(t) = -m(t + \Delta t) - m(t). \]  

(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.

\[ R(t) \equiv \frac{m(t)}{\alpha}. \]  

(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 \( R_{0,0} \), 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

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

(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 \( \alpha \), setting the partial derivatives to zero, and rearranging these terms, we can solve this type of nonlinear least square problems. We obtain

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

(12)

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

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

(13)

Next, we have

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

(14)

and

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

(15)

The other parameters \( A \) & \( \alpha \) 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

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

(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

\[ L \equiv \prod_{i=1}^{n} \left\{ \frac{(m(t_i) - m(t_{i-1}))}{(m_i - m_{i-1})! \exp [- (m(t_i) - m(t_{i-1}))]} \right\} \]  

(17)

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

\[ \ln L = \sum_{i=1}^{n} \left( m_i - m_{i-1} \right) \ln [m(t_i) - m(t_{i-1})] \]  

\[ - \sum_{i=1}^{n} \left( m(t_i) - m(t_{i-1}) \right) - \sum_{i=1}^{n} \ln [m_i - m_{i-1})!]. \]  

(18)
From (3), we know that \( m(t_i) - m(t_{i-1}) = a(\exp[-rW(t_{i-1})] - \exp[-rW(t_i)]) \). Thus,

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

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

\[
\frac{\partial \ln L}{\partial a} = \frac{\partial \ln L}{\partial r} = 0.
\] (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

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

\[
\sum_{i=1}^{N} v_i a_i \exp[-r_i W_i(t)] = \sum_{i=1}^{N} v_i a_i \exp[-r_i W_i^*].
\] (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 \( j \) is allocated \( W_j \) testing efforts; then the optimization problem can be represented as [11]-[14], [17], [18], [27]

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

Subject to:

\[
\sum_{i=1}^{N} W_i \leq W, \quad W_i \geq 0 \tag{24}
\]

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

From (25), we can obtain

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

Let \( D_i \equiv (-1/r_i) \ln[1 - R_0], \quad i = 1, 2, \ldots, N \). Thus, we have \( \sum_{i=1}^{N} W_i \leq W, \quad W_i \geq 0, \quad i = 1, 2, \ldots, N \), and \( W_i \geq C_i, \) 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]

\[
\text{Minimize}: \sum_{i=1}^{N} v_i a_i \exp[-r_i W_i] \tag{27}
\]

Subject to:

\[
\sum_{i=1}^{N} W_i \leq W, \quad W_i \geq 0 \text{ and } W_i \geq C_i. \tag{28}
\]

Let \( X_i = W_i - C_i \), then we can transform the above equations to

**Minimize:** \( \sum_{i=1}^{N} v_i a_i \exp[-r_i C_i] \exp[-r_i X_i] \) (29)

Subject to:

\[
\sum_{i=1}^{N} X_i \leq W - \sum_{i=1}^{N} C_i, \quad X_i \geq 0, \quad i = 1, 2, \ldots, N. \tag{30}
\]
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{equation}
\frac{\partial L(X_1, X_2, \ldots, X_N, \lambda)}{\partial X_i} = \sum_{i=1}^{N} v_i a_i \exp[-r_i C_i] \exp[-r_i X_i] + \lambda N X_i - W + \sum_{i=1}^{N} C_i = 0, \quad i = 1, 2, \ldots, N.
\end{equation}

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]

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

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

A3: \( \lambda \times \left[ \sum_{i=1}^{N} X_i - \left( N \sum_{i=1}^{N} C_i \right) \right] = 0, \quad i = 1, 2, \ldots, N. \)

**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) \( X_i \times \left[ N \sum_{i=1}^{N} C_i \right] = 0. \)

**Proof:**

a) From (30), we have \( \frac{\partial L(X_1, X_2, \ldots, X_N, \lambda)}{\partial X_i} = -v_i a_i r_i \exp[-r_i C_i] \times \exp[-r_i X_i] + \lambda. \) 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 \( X_i \times \left[ \lambda - (v_i a_i r_i \exp[-r_i C_i] \times \exp[-r_i X_i]) \right] = 0. \)

**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 \( X_i = \left\{ \ln(v_i a_i r_i \exp[-r_i C_i]) - \ln \lambda \right\} / \lambda. \)

**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 \( X_i \times (v_i a_i r_i \exp[-r_i C_i]) - v_i a_i r_i \exp[-r_i X_i] \times \exp[-r_i C_i] = 0 \) or \( X_i \times (v_i a_i r_i \exp[-r_i C_i]) \times (1 - \exp[-r_i X_i]) = 0. \) Because \( v_i \neq 0 \) or \( a_i \neq 0 \) or \( r_i \neq 0, \) 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 \( \lambda \geq v_i a_i r_i \exp[-r_i C_i] \times \exp[-r_i X_i] \) (because \( \exp[-r_i X_i] \leq 1 \)) or

\[ \lambda = v_i a_i r_i \exp[-r_i C_i] \times \exp[-r_i X_i] \neq 0. \]

Thus, the solution \( X^*_i \) is

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

The solution \( \lambda^0 \) is

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

Hence, we get \( X^0 = (X^*_1, X^*_2, \ldots, X^*_N) \) as an optimal solution to (30). However, the above \( X^0 \) may have some negative components if \( v_i a_i r_i \exp[-r_i C_i] < \lambda^0 \), 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

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

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

**Step 3:** Rearrange the index \( i \) such that

\( X^*_i \geq X^*_i \geq \ldots \geq X^*_N. \)

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

Else, \( X^*_{N-l} = 0; \quad l = l + 1. \)

**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,
Thus, the value of the objective function given by (38) at the optimal solution \((X_1^*, X_2^*, \ldots, X_N^*)\) as

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

(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

Minimize:

\[
\sum_{i=1}^{N} W_i,
\]

(36)

Subject to:

\[
\begin{align*}
\sum_{i=1}^{N} v_i a_i \exp[-r_i W_i] &\leq Z, & W_i &\geq 0 \\
R_i(t) &= 1 - \exp[-r_i W_i] & R_i(t) &\geq R_0.
\end{align*}
\]

(37)

Similarly, from (38), we can obtain \(W_i\)

\[
W_i \geq \frac{-1}{r_i} \ln(1 - R_0), \quad i = 1, 2, \ldots, N.
\]

(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

Minimize:

\[
\sum_{i=1}^{N} (X_i + C_i),
\]

(40)

Subject to:

\[
\begin{align*}
\sum_{i=1}^{N} v_i a_i \exp[-r_i C_i] \exp[-r_i X_i] &\leq Z, \\
X_i + C_i &\geq 0.
\end{align*}
\]

(41)

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

Minimize:

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

(42)

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

\[
\begin{align*}
\lambda &\leq \exp[r_i (C_i + X_i)] / v_i a_i r_i, \\
X_i &\times \{ 1 - \lambda v_i a_i r_i \exp[-r_i C_i] \exp[-r_i X_i] \} = 0.
\end{align*}
\]

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 \(1 - \lambda v_i a_i r_i \exp[-r_i C_i] \exp[-r_i X_i] \geq 0,\)

i.e., \(\lambda v_i a_i r_i \exp[-r_i C_i] \exp[-r_i X_i] \leq 1.\) Therefore, \(\lambda \leq \exp[r_i (C_i + X_i)] / v_i a_i r_i.\)

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)

\[
\begin{align*}
X_i &\geq 0 \text{ if } \lambda \leq \exp[r_i (C_i)] / v_i a_i r_i; \\
X_i &\times \{ 1 - \lambda v_i a_i r_i \exp[-r_i C_i] \exp[-r_i X_i] \} = 0.
\end{align*}
\]

Proof:

a) If \(X_i = 0\), then Theorem 2 part a) implies that \(\lambda \leq \exp[r_i (C_i)] / v_i a_i r_i.\) 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 \(\exp[-r_i X_i] \leq 1,\) then we have \(\lambda v_i a_i r_i \exp[-r_i C_i] \exp[-r_i X_i] \leq 1 or 1 - \lambda v_i a_i r_i \exp[-r_i C_i] \exp[-r_i X_i] \neq 0.\) 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

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

(43)

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

(44)

Thus, the solution \(X_i^0\) is

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

(45)
The solution $X^0$ is

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

That is

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

Hence, we get $X^0 = (X^0_1, X^0_2, X^0_3, \ldots, X^0_N)$ as an optimal solution to (42). However, the above $X^0$ may have some negative components if $v_i a_i r_i \exp[-r_i C_i] < \left( \frac{Z}{\sum_{i=1}^{N} \frac{1}{r_i}} \right)$, 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

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

**Step 3:** Rearrange the index $i$ such that $X^*_i \geq X^*_2 \geq \cdots \geq X^*_N - l$.

**Step 4:** If $X^*_i \geq 0$ then stop. Else update $X^*_i = 0$; $l = l + 1$. End If.

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

The optimal solution has the form

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

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

### 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 $a_i$, $r_i$, $\kappa$, and $v_i$, the optimal TE expenditures should be re-estimated. Therefore, we can know how much

### Table I

<table>
<thead>
<tr>
<th>Module</th>
<th>$a_i$</th>
<th>$r_i$</th>
<th>$\kappa$</th>
<th>$v_i$ in Example 1</th>
<th>$v_i$ in Example 2</th>
<th>$v_i$ in Example 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>89</td>
<td>4.18 x 10^4</td>
<td>1</td>
<td>1.0</td>
<td>1.0</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td>5.09 x 10^4</td>
<td>1</td>
<td>1.5</td>
<td>0.6</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>27</td>
<td>3.96 x 10^4</td>
<td>1</td>
<td>1.3</td>
<td>0.7</td>
<td>0.7</td>
</tr>
<tr>
<td>4</td>
<td>45</td>
<td>2.30 x 10^4</td>
<td>1</td>
<td>0.5</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>5</td>
<td>39</td>
<td>2.53 x 10^4</td>
<td>1</td>
<td>2.0</td>
<td>1.0</td>
<td>1.5</td>
</tr>
<tr>
<td>6</td>
<td>39</td>
<td>1.72 x 10^4</td>
<td>1</td>
<td>0.3</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>7</td>
<td>59</td>
<td>8.82 x 10^2</td>
<td>1</td>
<td>1.7</td>
<td>0.5</td>
<td>0.6</td>
</tr>
<tr>
<td>8</td>
<td>68</td>
<td>7.27 x 10^3</td>
<td>1</td>
<td>1.3</td>
<td>0.6</td>
<td>0.6</td>
</tr>
<tr>
<td>9</td>
<td>37</td>
<td>6.82 x 10^2</td>
<td>1</td>
<td>1.0</td>
<td>0.1</td>
<td>0.9</td>
</tr>
<tr>
<td>10</td>
<td>14</td>
<td>1.53 x 10^4</td>
<td>1</td>
<td>1.0</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

### Table II

<table>
<thead>
<tr>
<th>Module</th>
<th>$X^*_i$ for Example 1</th>
<th>$X^*_i$ for Example 2</th>
<th>$X^*_i$ for Example 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6254</td>
<td>8105</td>
<td>6015</td>
</tr>
<tr>
<td>2</td>
<td>3826</td>
<td>3547</td>
<td>2833</td>
</tr>
<tr>
<td>3</td>
<td>4117</td>
<td>4409</td>
<td>4052</td>
</tr>
<tr>
<td>4</td>
<td>2791</td>
<td>5191</td>
<td>4402</td>
</tr>
<tr>
<td>5</td>
<td>7825</td>
<td>8145</td>
<td>9030</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>403</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>13366</td>
<td>8267</td>
<td>8280</td>
</tr>
<tr>
<td>8</td>
<td>11820</td>
<td>11833</td>
<td>9343</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>6046</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
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 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.

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 cases of 40%, 30%, 20%, and 10% increase in \( a_1 \) (Algorithm 1).

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)

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

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)

<table>
<thead>
<tr>
<th>Module</th>
<th>$X^r_1(a_1=0.6)$</th>
<th>$X^r_1(a_1=0.7)$</th>
<th>$X^r_1(a_1=0.8)$</th>
<th>$X^r_1(a_1=0.9)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5105</td>
<td>5452</td>
<td>5752</td>
<td>6017</td>
</tr>
<tr>
<td>2</td>
<td>3886</td>
<td>3768</td>
<td>3852</td>
<td>3838</td>
</tr>
<tr>
<td>3</td>
<td>4194</td>
<td>4171</td>
<td>4151</td>
<td>4133</td>
</tr>
<tr>
<td>4</td>
<td>2923</td>
<td>2883</td>
<td>2849</td>
<td>2818</td>
</tr>
<tr>
<td>5</td>
<td>7945</td>
<td>7909</td>
<td>7877</td>
<td>7850</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>13710</td>
<td>13606</td>
<td>13516</td>
<td>13437</td>
</tr>
<tr>
<td>8</td>
<td>12237</td>
<td>12112</td>
<td>12003</td>
<td>11906</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Module</th>
<th>$X^r_1(a_1=0.6 &amp; a_2=0.6)$</th>
<th>$X^r_1(a_1=0.7 &amp; a_2=0.7)$</th>
<th>$X^r_1(a_1=0.8 &amp; a_2=0.8)$</th>
<th>$X^r_1(a_1=0.9 &amp; a_2=0.9)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5094</td>
<td>5338</td>
<td>5609</td>
<td>5912</td>
</tr>
<tr>
<td>2</td>
<td>3886</td>
<td>3873</td>
<td>3859</td>
<td>3844</td>
</tr>
<tr>
<td>3</td>
<td>4195</td>
<td>4178</td>
<td>4160</td>
<td>4140</td>
</tr>
<tr>
<td>4</td>
<td>2925</td>
<td>2897</td>
<td>2865</td>
<td>2830</td>
</tr>
<tr>
<td>5</td>
<td>7946</td>
<td>7921</td>
<td>7892</td>
<td>7861</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>13713</td>
<td>13640</td>
<td>13559</td>
<td>13468</td>
</tr>
<tr>
<td>8</td>
<td>12241</td>
<td>12153</td>
<td>12055</td>
<td>11944</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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 in $r_1$ (ALGORITHM 1). Table X shows numerical values of the optimal TE expenditures for the cases of 40%, 30%, 20%, and 10% increase in $r_1$ & $r_2$ (ALGORITHM 1). Finally,
Similarly, we investigate the possible change of module is decreased. From Table XIV, it are changed. For the data set (Example 1) used in is decreased by 30%, the estimated value of is low. From Table XVI, is increased by 40%, then the estimated is low. Next, we show the same comparison results in case that is low. For each software module, then we can calculate the MTEE and 1 & 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 $a_1$ (Algorithm 2). 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 10.4% increment). Be- sides, 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%, 4.2%, 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).

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 $a_1$ & $a_i$ $(i \neq 1)$. If we let $a_1$ & $a_2$ both be increased by 40%, then the estimated values of optimal TE expenditure for modules 1, 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, 2 are changed from 7700 to 6847 (about 11.1% decrement), and

**Table XII**

<table>
<thead>
<tr>
<th>Module</th>
<th>$X_i'$</th>
<th>$X_i'$</th>
<th>$X_i'$</th>
<th>$X_i'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8504</td>
<td>8327</td>
<td>8316</td>
<td>7928</td>
</tr>
<tr>
<td>2</td>
<td>5013</td>
<td>5013</td>
<td>5013</td>
<td>5013</td>
</tr>
<tr>
<td>3</td>
<td>5643</td>
<td>5643</td>
<td>5643</td>
<td>5643</td>
</tr>
<tr>
<td>4</td>
<td>5424</td>
<td>5424</td>
<td>5424</td>
<td>5424</td>
</tr>
<tr>
<td>5</td>
<td>10211</td>
<td>10211</td>
<td>10211</td>
<td>10211</td>
</tr>
<tr>
<td>6</td>
<td>1770</td>
<td>1770</td>
<td>1770</td>
<td>1770</td>
</tr>
<tr>
<td>7</td>
<td>20220</td>
<td>20220</td>
<td>20220</td>
<td>20220</td>
</tr>
<tr>
<td>8</td>
<td>20131</td>
<td>20131</td>
<td>20131</td>
<td>20131</td>
</tr>
<tr>
<td>9</td>
<td>7759</td>
<td>7759</td>
<td>7759</td>
<td>7759</td>
</tr>
<tr>
<td>10</td>
<td>2388</td>
<td>2388</td>
<td>2388</td>
<td>2388</td>
</tr>
</tbody>
</table>

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

**Table XIII**

<table>
<thead>
<tr>
<th>Module</th>
<th>$X_i'(a_1=1.4)$</th>
<th>$X_i'(a_1=1.3)$</th>
<th>$X_i'(a_1=1.2)$</th>
<th>$X_i'(a_1=1.1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8504</td>
<td>8327</td>
<td>8316</td>
<td>7928</td>
</tr>
<tr>
<td>2</td>
<td>5013</td>
<td>5013</td>
<td>5013</td>
<td>5013</td>
</tr>
<tr>
<td>3</td>
<td>5643</td>
<td>5643</td>
<td>5643</td>
<td>5643</td>
</tr>
<tr>
<td>4</td>
<td>5424</td>
<td>5424</td>
<td>5424</td>
<td>5424</td>
</tr>
<tr>
<td>5</td>
<td>10211</td>
<td>10211</td>
<td>10211</td>
<td>10211</td>
</tr>
<tr>
<td>6</td>
<td>1770</td>
<td>1770</td>
<td>1770</td>
<td>1770</td>
</tr>
<tr>
<td>7</td>
<td>20220</td>
<td>20220</td>
<td>20220</td>
<td>20220</td>
</tr>
<tr>
<td>8</td>
<td>20131</td>
<td>20131</td>
<td>20131</td>
<td>20131</td>
</tr>
<tr>
<td>9</td>
<td>7759</td>
<td>7759</td>
<td>7759</td>
<td>7759</td>
</tr>
<tr>
<td>10</td>
<td>2388</td>
<td>2388</td>
<td>2388</td>
<td>2388</td>
</tr>
</tbody>
</table>

**Table XIV**

<table>
<thead>
<tr>
<th>Module</th>
<th>$X_i'(a_1=0.6, a_2=0.6)$</th>
<th>$X_i'(a_1=0.7, a_2=0.7)$</th>
<th>$X_i'(a_1=0.8, a_2=0.8)$</th>
<th>$X_i'(a_1=0.9, a_2=0.9)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6478</td>
<td>6847</td>
<td>7166</td>
<td>7448</td>
</tr>
<tr>
<td>2</td>
<td>4010</td>
<td>4313</td>
<td>4575</td>
<td>4806</td>
</tr>
<tr>
<td>3</td>
<td>5643</td>
<td>5643</td>
<td>5643</td>
<td>5643</td>
</tr>
<tr>
<td>4</td>
<td>5424</td>
<td>5424</td>
<td>5424</td>
<td>5424</td>
</tr>
<tr>
<td>5</td>
<td>10211</td>
<td>10211</td>
<td>10211</td>
<td>10211</td>
</tr>
<tr>
<td>6</td>
<td>1770</td>
<td>1770</td>
<td>1770</td>
<td>1770</td>
</tr>
<tr>
<td>7</td>
<td>20220</td>
<td>20220</td>
<td>20220</td>
<td>20220</td>
</tr>
<tr>
<td>8</td>
<td>20131</td>
<td>20131</td>
<td>20131</td>
<td>20131</td>
</tr>
<tr>
<td>9</td>
<td>7759</td>
<td>7759</td>
<td>7759</td>
<td>7759</td>
</tr>
<tr>
<td>10</td>
<td>2388</td>
<td>2388</td>
<td>2388</td>
<td>2388</td>
</tr>
</tbody>
</table>

**Table XV**

<table>
<thead>
<tr>
<th>Module</th>
<th>$X_i'(a_1=1.4, a_2=1.4)$</th>
<th>$X_i'(a_1=1.3, a_2=1.3)$</th>
<th>$X_i'(a_1=1.2, a_2=1.2)$</th>
<th>$X_i'(a_1=1.1, a_2=1.1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8504</td>
<td>8327</td>
<td>8316</td>
<td>7928</td>
</tr>
<tr>
<td>2</td>
<td>5013</td>
<td>5013</td>
<td>5013</td>
<td>5013</td>
</tr>
<tr>
<td>3</td>
<td>5643</td>
<td>5643</td>
<td>5643</td>
<td>5643</td>
</tr>
<tr>
<td>4</td>
<td>5424</td>
<td>5424</td>
<td>5424</td>
<td>5424</td>
</tr>
<tr>
<td>5</td>
<td>10211</td>
<td>10211</td>
<td>10211</td>
<td>10211</td>
</tr>
<tr>
<td>6</td>
<td>1770</td>
<td>1770</td>
<td>1770</td>
<td>1770</td>
</tr>
<tr>
<td>7</td>
<td>20220</td>
<td>20220</td>
<td>20220</td>
<td>20220</td>
</tr>
<tr>
<td>8</td>
<td>20131</td>
<td>20131</td>
<td>20131</td>
<td>20131</td>
</tr>
<tr>
<td>9</td>
<td>7759</td>
<td>7759</td>
<td>7759</td>
<td>7759</td>
</tr>
<tr>
<td>10</td>
<td>2388</td>
<td>2388</td>
<td>2388</td>
<td>2388</td>
</tr>
</tbody>
</table>
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**


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’96, 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.