A Software Cost Model with Reliability Constraint under Two Operational Scenarios

Satoru UKIMOTO and Tadashi DOHI

Department of Information Engineering, Graduate School of Engineering
Hiroshima University, 1–4–1 Kagamiyama, Higashi-Hiroshima, 739–8527 Japan
dohi@rel.hiroshima-u.ac.jp

Abstract. In this paper we extend the reliability constrained cost minimization (RCCM) model by Helander *et al.* (1998) from two view points: time non-homogeneous property on software failure-occurrence process and gap between testing and operational phases of software product. The expected cost minimizations with reliability constraint are formulated as non-linear minimization problems under alternative scenario on operation. We analytically develop optimization algorithms for the optimal allocation policies.

Key words: software cost model, reliability constraint, operational phase, non-homogeneous Poisson process, non-linear optimization.

1 Reliability Constrained Cost Minimization

We introduce RCCM (Reliability Constrained Cost Minimization) model by He-lander *et al.* [1]. Suppose that software consists of n components whose failure intensity functions are given by A_i (i = 1, 2,.. . , n). More precisely, let $\{N_i(t), t \sim 0\}$ be the cumulative number of software failures occurred by time t in the i-th component and be a homogeneous Poisson process (HPP), where the intensity of an HPP, A_i, denotes the instantaneous software failure-occurrence rate per each failure for i-th component. The mean value functions $E[N_i(t)] = A_i t$ and $E[\sum_{i=1}^{n} N_i(t)] = \sum_{i=1}^{n} A_i t$ denote the cumulative numbers of software failures occurred in the i-th component and the whole software system by time t, respectively.

Let $TC(A_1, A_2,.. . , A_n; r)$ and $R(A_1, A_2,.. .. , A_n; r)$ be the expected total operational cost and the quantitative software reliability, respectively, as functions of software intensity A_i (i = 1, 2,. . . , n), where r (> 0) denotes the operational period of software after the release. Suppose that the expected total operational cost is given by the sum of individual expected operational cost, $C_i(A_i)$, for each software component i (= 1, 2,. . . , n):

$$TC(A_1,A_2,...,A_n;r) = \sum_{i=1}^{n} C_i(A_i;r). \quad (1)$$

From the property of expected operational cost for each component i, we assume that the function $C_i(A_i; \tau)$ is monotonically decreasing and convex in A_i without any loss of generality. Helander *et al.* [1] assume thee kinds of cost functions; (i) linear function $C_i(A_i; \tau) = -\alpha_i A_i \tau + \beta_i$, (ii) logarithmic exponential (LogExp) function $C_i(A_i; \tau) = \beta_i \ln(1 - \exp[-A_i\tau])$, and (iii) inverse power (InvPow) function $C_i(A_i; \tau) = \beta_i/(A_i\tau - S_i)^{\square i}$, for i = 1, 2, ... , n, where α_i (> 0), β_i (> 0) and S_i (> 0) are constant.

Then the RCCM with a fixed τ is formulated as the following minimization problem with respect to the software intensity:

$$\min_{\square i;\, i=1,2,\ldots,v} \quad TC(A1, A2, \ldots, A_v; \tau),$$

$$s.t. R(A1, A2, \ldots, A_v; \tau) > \rho,\ A_i > 0 \text{ for } i = 1, \ldots n, \qquad (2)$$

where ρ in Eq.(3) is the minimum requirement of quantitative software reliability level and $R(A1, A2, \ldots, A_v; \tau)$ means the probability that the software does not fail during the operational period τ.

It is common to see that the cumulative number of software failures caused by software faults is estimated by the operational profile [2] inferred in the design phase. Let ϕ_i be the frequency to use the component i in the operational phase with $E_{i=v} \phi_i = 1$. Similar to Helander *et al.* [1], let $\mu_{i\varphi}$ and p_φ denote the ratio of the j-th software operation (j = 1, 2, ... , m), processed by component i and the probability that the j-th software operation is executed, respectively, where $E \mu\ 1\ p_\varphi = 1$ and $E_i{}^{v} = 1\ \varphi = 1 \mu_{i\varphi} = 1$. Then the frequency to use the component $\varphi = 1$ is given by $\phi_i = \varphi = 1\ \mu_{i\varphi} p_\varphi$, so that the quantitative software reliability is derived by

$$R(A1, A2, ..., A_v;\tau) = \exp\{-\sum_{i=1}^{v} \} \phi_i A_i \tau \qquad (3)$$

from the HPP assumption. By solving the minimization problem in Eq.(3), the optimal software intensity for respective software components, AI (i = 1, 2, ... , n), can be obtained so as to satisfy the reliability constraint. For the minimization problem in Eq.(3) with inequality constraints, it is straightforward to get necessary conditions of optimality (Kuhn-Tucker conditions) and to characterize the optimal software intensity AI . We summarize the optimal software intensity AI for three models in Table 1, where u^* is the optimal Lagrange multiplier which is given as a unique solution of nonlinear equations.

2 Extended RCCM Model

The common experiences suggest that the software fault-detection process in testing is represented by a non-homogeneous Poisson process (NHPP). Suppose that the software failures are caused by software faults. Let $\{N_i(t), t > 0\}$ be the cumulative number of software faults detected by time t for i-th component and be an NHPP with intensity function $A_i(t; o_s)$. In the above definition,

Table 1. Optimal allocation policies.

Cost	HPP	NHPP
Linear	$a_i^* = 1 - \ln(\pi)/100$ (i = k); 0 (i ≠ k)	$a_i^* = 1 - \ln(\pi)/100$ (i = k); 0 (i ≠ k)
LogExp	$\lambda_i^* \ln + \ln(\rho) + \sum_{i=1}^{n} \phi_i \lambda_i^* \tau = 0$	$a_i^* = \ln(\rho) + \sum_{i=1}^{n} a_i^* F(\tau) = 0$
InvPow	$\lambda_i^* = \ln(\rho) + \sum_{i=1}^{n} \phi_i \lambda_i^* \tau = 0$	$a_i^* = \ln(\rho) + \sum_{i=1}^{n} a_i^* F(\tau) = 0$

$\Theta = (\Theta_1, \Theta_2, \ldots, \Theta_n)$ is the model parameters (vector) included in each intensity function Θ_i ($i = 1, 2, \ldots, n$). Then the p.m.f. of the total cumulative number of software faults detected in testing is given by

$$\Pr\{N(t) = x\} = \Pr\{\sum_{i=1}^{n} N_i(t) = x\} = \frac{\{\Lambda(t;\Theta)\}^x}{x!} \exp[-\Lambda(t;\Theta)], \tag{4}$$

$$\Lambda(t;\Theta) = \sum_{i=1}^{n} \Lambda_i(t;\Theta_i) = \sum_{i=1}^{n} \int_0^t \lambda_i(x;\Theta_i)dx, \tag{5}$$

where the functions $\Lambda(t;\Theta) = \mathrm{E}[N(t)]$ and $\Lambda_i(t;\Theta_i) = \mathrm{E}[N_i(t)]$ are called the mean value functions. For the two-parameters case with $\Theta_i = (a_i, b_i)$ ($i = 1, 2, \ldots, n$), the mean value functions have the product forms of $\Lambda_i(t;\Theta_i) = a_i F_i(t; b_i)$, where $F_i(t; b_i)$ are the cumulative distribution functions (c.d.f.) of independent and identically distributed fault-detection times in the i-th component, and $a = \sum_{i=1}^{n} a_i$ is the expected initial number of software faults remaining before testing, i.e., $\lim_{t\to 0} \Lambda(t;\Theta) = a$ and $\lim_{t\to 0} \Lambda_i(t;\Theta_i) = \lim_{t\to 0} F_i(t; b_i) = a_i$.

Define the software debug rate in each component

$$d_i(t) = \frac{d\Lambda_i(t; a_i, b_i)/dt}{\Lambda_i(\infty; a_i, b_i) - \Lambda_i(t; a_i, b_i)} = f_i(t;b_i)/\{1 - F_i(t; b_i)\}, \tag{6}$$

where the function $f_i(t; b_i) = dF_i(t; b_i)/dt$ is the probability density function (p.d.f.) of $F_i(t; b_i)$, so that $d_i(t)$ is equivalent to the failure rate of the c.d.f. $F_i(t; b_i)$ and is independent of a_i. It should be noted that the NHPP-based model describes the debugging process in the software testing. On the other hand, the RCCM focuses on the operational phase after the release of software, but does not link to the testing phase.

Yang and Xie [3] independently consider a scenario to change from software testing phase to operational phase, and assume that the software intensity function $\lambda(t;\Theta) = \sum_{i=1}^{n} \lambda_i(t;\Theta_i)$ becomes a constant just after releasing the software.

For a given release time t_0, the software intensity is give as $A_i(t_0; \square i) = A_i$. In other words, the RCCM model [1] implicitly assumes the above scenario in the operational phase and controls the software intensity, A_i (i = 1, 2, ... , n), in each component. We call this scenario that the software intensity becomes a constant after releasing the software *Scenario 1*.

In this paper we propose alternative scenario, *Scenario 2*, where the software debug rate $d_i(t)$ in Eq.(16), but not the software intensity function $A_i(t)$, becomes a constant after the release, say, $d_i(t_0)$. This implies that the software failure-occurrence time distribution $F_i(t; b_i)$ after the release time t_0 ($< t$) in the i-th component is an exponential distribution with rate $d_i(t_0)$, say $b_i = d_i(t_0)$ and $F_i(t;b_i) = a_i\{1 - \exp(-d_i(t_0)t)\}$ iff the software debug rate is constant for t ($> t_0$). Then, we have

$$\Lambda_i(t; \square i) = \begin{cases} a_i F(t; b_i) & (0 < t < t_0) \\ a_i F_i(t; d_i(t_0)) & \\ \quad + a_i\{1 - F_i(t_0; d_i(t_0))\}\{1 - e^{-d_i(t_0)[t-t_0]}\}. & (t_0 < t < t_0 + \tau), \end{cases} \tag{7}$$

It seems that Scenario 1 with constant intensity level in the operational phase leads to the linear increase of cumulative number of failures and does not reflect the testing effort spent in the testing. On the other hand, Scenario 2 implies that the software intensity changes to a specific exponential form with rate $d_i(t_0)$ but results the consistent behavior of the cumulative number of failures.

Based on Scenario 2, we re-formulate RCCM model. Instead of the software intensity A_i in the original RCCM model, we regard the expected initial number of faults in each component a_i as the decision variable. Note that a_i is a target variable in the software testing, so that the software testing manager will set the objective number of faults detected in each software component. For given $d_i(t_0) = d_i$, (i = 1, 2, ... , n), we formulate the extended RCCM by

$$\min_{a_i; i=1,2,\ldots,n} \quad TC(a_1, a_2, \ldots, a_i; \tau)$$

$$\text{s.t.} R(a_1, a_2, \ldots, a_n; \tau) > \rho,\ a_i > 0 \text{ for } i = 1, 2, \ldots n, \tag{8}$$

where

$$R(a_1, a_2, \ldots, a_i; \tau) = \exp\left[-\sum_{i=1}^{n}\right] a_i(1 - \exp[-d_i(t_0)\phi_i\tau]) . \tag{9}$$

Solving the Kuhn-Tucker conditions for three cost functions, we also summarize the optimal allocation poicy $a^{\square}_i$ (i = 1, 2, ... , n) in Table 1, where $F(\tau) = 1 - \exp[-d_i(t_0)\phi_i\tau]$.

References

1. Helander, M. E., Zhao, M. and Ohlsson, N.: Planning models for software reliability and cost. *IEEE Transactions on Software Engineering* 24 (6), 420–434 (1998).
2. Musa, J. D.: Operational profiles in software-reliability engineering. *IEEE Software*, 10 (2), 14–32 (1993).
3. Yang, B. and Xie, M.: A study of operational and testing reliability in software reliability analysis. *Reliability Engineering & System Safety*, 70 (1), 323–329 (2000).