Multi-criteria Decision Making in Optimal Software Testing-Allocation Problem

Shinji Inoue, Yuka Minamino, and Shigeru Yamada

Abstract We discussed estimating optimal testing-resource allocation for the module testing in software testing phase by applying the notion of the multi-attribute utility theory. Concretely, considering the utilities of software development manager for the reliability, testing-resource and testing-cost, we define testing-resource allocation problems for estimating optimal testing-resource allocation maximizing the utility of the software development manager under the certain testing-strategy. Finally, we show examples of the applications of our approaches by using actual data, and give some consideration on our results and the importance of developing testing management strategy in the software module testing.

1 Introduction

It is well-known that software testing located in the software development process is resource-consuming process. The testing-resource means CPU time, man-month, the number of test cases, and so forth. Generally, software testing consists of the module, integration and system testing. Especially in the module testing, a lot of testing-resource is needed to enhance software reliability of each software module because the software modules are tested independently. Therefore, it is important for software development managers to allocate the testing-resource in the module testing efficiently. The optimal testing-resource allocation problem (Yamada and Ohtera 1990; Nishiwaki et al. 1995; Yamada et al. 1995; Ichimori et al. 2002) is known as one of the interesting problems for software development management.

S. Inoue (✉)
Kansai University, 2-1-1, Ryozenji-cho, Takatsuki-shi, Osaka 569-1095, Japan
e-mail: ino@kansai-u.ac.jp

Y. Minamino · S. Yamada
Tottori University, 4-101, Minami, Koyama-cho, Tottori-shi, Tottori 680-8552, Japan
e-mail: minamino@tottori-u.ac.jp

S. Yamada
e-mail: yamada@tottori-u.ac.jp

A. G. Aggarwal et al. (eds.), *Optimization Models in Software Reliability*,
Springer Series in Reliability Engineering,
https://doi.org/10.1007/978-3-030-78919-0_4

The optimal testing-resource allocation problems discuss how to allocate the testing-resource for each module testing to conduct debugging activities more efficiently. As for most of existing discussions on optimal testing-resource allocation problem, the optimal testing-resource allocation is decided by minimizing the total number of remaining faults in the all software modules under certain constraint of the total testing-resource for conducting the whole module testing.

However, from the point of view of actual software development management, it must be better to consider several evaluation criteria, not only the constraint of the total testing-resource, for deciding the testing-resource allocation in the module testing. As one of the approaches, we discuss optimal testing-resource allocation problems based on the multi-attribute utility theory. The multi-attribute utility theory has been often applied in discussion of software development management. Especially, Kapur et al. (2012) applied the multi-attribute utility theory for developing an optimal software release problem and discussed the estimation procedure for estimating optimal software release time maximizing the utility of the software development manager. We discuss an application of multi-attribute utility theory to the testing-resource allocation problem, which is one of the interesting issues on software development management. Compared with the existing discussion on the optimal testing-resource allocation approach (Nishiwaki et al. 1995; Yamada et al. 1995; Ichimori et al. 2002), our approach gives the another aspects in the software develop management by proposing another optimal problem for testing-resource allocation, such as multi-attribute maximization problem on estimating the optimal testing-resource allocation in the module testing. Further, we show examples of the applications of our approaches by using actual data, in which the multi-attribute utility is formulated from the aspects of the reliability, testing-resource, and testing-cost. And we give some considerations on our results and the importance to develop testing management strategy in the module testing.

2 Section Heading

We introduce an existing approach for optimal testing-resource allocation problem (Nishiwaki et al. 1995; Yamada et al. 1995; Ichimori et al. 2002). In this approach, the software reliability growth process for each module is described by using the testing-effort dependent software reliability growth model (Yamada 2014; Yamada and Ohtera 1990). Let us denote the mean value function of the nonhomogeneous Poisson process (Pham 2000) or the expected number of faults detected up to testing time t by $Z(t)$. The testing-effort dependent software reliability growth model is given as

$$Z(t) = a(1 - \exp[-r \cdot TE(t)]), \tag{1}$$

which is the mean value function of the non-homogeneous Poisson process. In Eq. (1), $TE(t)$ is the total testing-effort expended up to time t, a is the initial faults content,

and r is the software fault detection rate per expended testing-effort. Regarding the function $TE(t)$ in Eq. (1), we have

$$TE(t) = \int_0^t te(t)\mathrm{d}t, \tag{2}$$

where $te(t)$ is the testing-effort expenditure expended at testing-time t. And the expected number of remaining faults, $n(t)$, is given as

$$n(t) = a - Z(t) = a \cdot \exp[-r \cdot TE(t)]. \tag{3}$$

The optimal testing-recourse allocation problem is discussed within the following situation: (1) a software system consists of M independent software modules, (2) the number of remaining faults in each module can be estimated by the testing-effort dependent software reliability growth model, (3) the software development managers have to allocate the testing-resource expenditures to each software module testing efficiently, so that the total number of remaining faults in the software system may be minimized.

Let d_i denote the amount of testing-resource expenditure spent for testing software module, i ($i = 1, 2, \cdots, M$). From Eq. (3), the expected number of remaining faults in the software module i, which is denoted by n_i is given as

$$n_i = a_i \cdot \exp[-r_i d_i], \tag{4}$$

where a_i is the initial fault content for the software module i and r_i represents the fault detection rate per expended testing-resource for the software module i ($0 < r_i < 1$). Then, the total expected number of remaining faults in the software system is

$$N = \sum_{i=1}^{M} n_i. \tag{5}$$

Then, the software testing-resource allocation problem is formulated as

$$\left.\begin{aligned} &\text{min: } \sum_{i=1}^{M} \delta_i a_i \cdot \exp[-r_i d_i] \\ &\text{subject to } \sum_{i=1}^{M} d_i \le W_P, \quad d_i \ge 0, \end{aligned}\right\}, \tag{6}$$

where W_P is the planned total amount of the testing-resource and δ_i is the weight representing the importance or complexity of the software module i. We should note that the optimal testing-resource allocation problem in Eq. (6) is defined as the minimization problem on the number of remaining faults in the whole software modules under the constraint of the planned total amount of the testing-resource. For

solving the Eq. (6), the Lagrange multipliers method is generally applied. Then, we have

$$L = \sum_{i=1}^{M} \delta_i a_i \cdot \exp[-r_i d_i] + \lambda \left(\sum_{i=1}^{M} d_i - W_P \right), \tag{7}$$

where λ is the Lagrange multiplier. The necessary and sufficient conditions for obtaining the optimal solutions are

$$\left.\begin{aligned} &\frac{\partial L}{\partial d_i} = -\delta_i a_i r_i \cdot \exp[-r_i d_i] + \lambda = 0 \\ &\frac{\partial L}{\partial \lambda} = \sum_{i=1}^{M} d_i - W_P \geq 0 \\ &\lambda \left(\sum_{i=1}^{M} d_i - W_P \right) = 0, \qquad \lambda \geq 0 \end{aligned}\right\}. \tag{8}$$

Further, we assume $A_1 \geq A_2 \geq \cdots \geq A_{k-1} \geq \lambda \geq A_k \geq A_{k+1} \geq \cdots \geq A_M$, where $A_i = \delta_i a_i r_i$ $(i = 1, , 2, \cdots, M)$. Then, the optimal testing-resource allocation, d_i^*, is derived as

$$\left.\begin{aligned} &d_i^* = -\frac{1}{r_i}(\ln A_i - \ln \lambda) \qquad (i = 1, 2, \ldots, k-1), \\ &d_i^* = 0 \qquad (i = k, k+1, \ldots, M) \end{aligned}\right\}, \tag{9}$$

where $\ln \lambda$ is given as

$$\ln \lambda = \frac{\sum_{i=1}^{M} \frac{1}{r_i} \ln A_i - W_P}{\sum_{i=1}^{M} \frac{1}{r_i}}. \tag{10}$$

Consequently, the optimal testing-resource allocation is obtained as

$$d_i^* = \max \left\{ 0, -\frac{1}{r_i}(\ln A_i - \ln \lambda) \right\} \qquad (i = 1, 2, \ldots, M). \tag{11}$$

3 Proposed Approach

We propose another approaches for estimating optimal testing-resource allocation with multi-attribute utility of software development manager. Now, we consider the following situation:

(1) A software system consists of M independent software modules.

(2) The number of remaining faults for each software module are estimated by the testing-effort dependent software reliability growth model.
(3) The software development managers allocate the testing-resource to testing for each software module for maximizing their utility, which consists of several attributes being related to the software development management.

We now consider that the following attributes: the testing-resource and the software reliability attributes, respectively. Regarding the testing-resource attribute, we formulate

$$\min : E = \frac{\sum_{i=1}^{M} d_i}{W_P} = \frac{W}{W_P}, \tag{12}$$

since the software managers prefer to spend the testing-resource less than the planned total amount of testing-resource. In Eq. (12), W is the testing-resource expenditure spent up to the end of the module testing. Further, we give the software reliability attribute as

$$\max : R = 1 - \frac{\sum_{i=1}^{M} a_i \cdot \exp[-r_i d_i]}{\sum_{i=1}^{M} a_i} = 1 - \frac{N}{\sum_{i=1}^{M} a_i}. \tag{13}$$

Here we develop the utility function for each attribute based on the following certain situation on the software development management strategy: (1) for the testing-resource attribute, at least 60% of the planned total amount of testing-resources must be consumed, (2) for the software reliability attribute, at least 80% of software faults should be removed, (3) the software development managers take the risk neutral position for each attribute. Then, we set the lowest and highest consumptions for the testing-resource attribute are $E^L = 0.6$ and $E^H = 1.0$, respectively. And the lowest and highest requirements for the software reliability attribute are $R^L = 0.8$ and $R^H = 1.0$, respectively. Then, we obtain the following utility functions for each attribute based on the notion of the risk neutral position:

$$\left.\begin{array}{l} u(E) = 2.5E - 1.5 \\ u(R) = 5R - 4 \end{array}\right\}. \tag{14}$$

From Eq. (14), we define the following optimal testing-resource allocation problem with the additive multi-attribute utility function under the testing management strategy:

$$\left.\begin{array}{ll} \max : u(E, R) = \delta_R \cdot u(R) - \delta_E \cdot u(E) & \\ \qquad\qquad = \delta_R \cdot (5R - 4) - \delta_E \cdot (2.5E - 1.5) & \\ \text{subject to} \quad \delta_R + \delta_E = 1, \quad d_i \geq 0 & \end{array}\right\}, \tag{15}$$

where δ_R and δ_E are the weight parameters for the attributes R and E, respectively. Consequently, we obtain the optimal testing-resource allocation, d_i^* $(i =$

$1, 2, \ldots, M$), by maximizing the multi-attribute utility function in Eq. (15). From Eq. (15), the optimal testing-resource allocation, d_i^*, is obtained by

$$\left.\begin{array}{ll} d_i^* = -\dfrac{1}{r_i} \ln \dfrac{2.5\delta_E \sum_{i=1}^{M} a_i}{5\delta_R a_i r_i W_P} & (i = 1, 2, \cdots, k-1) \\ d_i^* = 0 & (i = k, k+1, \ldots, M) \end{array}\right\}, \tag{16}$$

Then, we have

$$d_i^* = \max\left\{0, -\frac{1}{r_i} \ln \frac{2.5\delta_E \sum_{i=1}^{M} a_i}{5\delta_R a_i r_i W_P}\right\} \qquad (i = 1, 2, \cdots, M). \tag{17}$$

In Eq. (17), we should note that

$$B_1 \geq B_2 \geq \cdots \geq B_{k-1} \geq \frac{2.5\delta_E \sum_{i=1}^{M} a_i}{5\delta_R W_P} \geq B_k \geq B_{k+1} \geq \cdots \geq B_M, \tag{18}$$

where $B_i = a_i r_i$ ($i = 1, 2, \ldots, M$).

Furthermore, we add the cost attribute for treating more practical situation. For formulating the cost attribute, we set the following cost parameters:

c_1: the debugging cost per fault discovered in the module testing,
c_2: the debugging cost per fault undiscovered during the module testing ($c_1 < c_2$)
c_3: the cost per unit of the testing-resource for the module testing.

Then, the cost function is given as

$$V = c_1 \sum_{i=1}^{M} a_i (1 - \exp[-r_i d_i]) + c_2 \sum_{i=1}^{M} a_i \exp[-r_i d_i] + c_3 \sum_{i=1}^{M} d_i, \tag{19}$$

by following the notion of the testing-effort dependent software reliability growth model. From Eq. (19), the cost attribute is given as

$$\text{min:} \quad C = \frac{V}{C_P}, \tag{20}$$

where C_P represents the planned budget for software testing. And we add the following test management strategy: (4) for the cost attribute, at least 50% of the budget must be consumed in the module testing. The lowest and highest consumptions for the cost attribute are $C^L = 0.5$ and $C^H = 1.0$. Further, we assume the following utility function on the cost attribute: $u(C) = 2C - 1$, which is just one of the examples.

When we consider the three attributes, such as the testing-resource, software reliability, and cost attributes, the optimal testing-resource allocation problem with the three attributes is defined as

$$\left.\begin{array}{ll} \max : u(E,R,C) = \delta_R \cdot u(R) - \delta_E \cdot u(E) - \delta_C \cdot u(C) & \\ \qquad\qquad\quad = \delta_R \cdot (5R-4) - \delta_E \cdot (2.5E-1.5) - \delta_C \cdot (2C-1) & \\ \text{subject to} \qquad \delta_R + \delta_E + \delta_C = 1, \quad d_i \geq 0 & \end{array}\right\}, \quad (21)$$

where δ_C is the weight parameter for the cost attribute. From Eq. (21), the optimal testing-resource allocation, d_i^*, is obtained by

$$\left.\begin{array}{ll} d_i^* = -\dfrac{1}{r_i} \ln \dfrac{\frac{2.5\delta_E}{W_P} + \frac{2\delta_C c_3}{C_P}}{a_i r_i \left\{ \frac{5\delta_R}{\sum_{i=1}^{M} a_i} + \frac{2(c_2-c_1)\delta_C}{C_P} \right\}} & (i = 1, 2, \ldots, k-1) \\ d_i^* = 0 \quad (i = k, k+1, \ldots, M) & \end{array}\right\}. \quad (22)$$

Then, we have

$$d_i^* = \max \left\{ 0, -\frac{1}{r_i} \ln \frac{\frac{2.5\delta_E}{W_P} + \frac{2\delta_C c_3}{C_P}}{a_i r_i \left\{ \frac{5\delta_R}{\sum_{i=1}^{M} a_i} + \frac{2(c_2-c_1)\delta_C}{C_P} \right\}} \right\} \quad (i = 1, 2, \ldots, M). \quad (23)$$

In Eq. (23), we should note

$$C_1 \geq C_2 \geq \cdots \geq C_{k-1} \geq \frac{\frac{2.5\delta_E}{W_P} + \frac{2\delta_C c_3}{C_P}}{\frac{5\delta_R}{\sum_{i=1}^{M} a_i} + \frac{2(c_2-c_1)\delta_C}{C_P}} \geq C_k \geq C_{k+1} \geq \cdots \geq C_M. \quad (24)$$

4 Numerical Examples

We show numerical examples for our proposed approach by using actual data Yamada and Ohtera (1990). The data consists of 10 modules and 251 faults still remain through the module testing. Table 1 shows the estimated values of a_i and r_i, which have been estimated by following the testing-effort dependent software reliability growth model. "M" means module, then "M1" means the module 1. We set the cost parameters as $c_1 = 1$, $c_2 = 2$, and $c_3 = 5$. And we also set the planned total amount of testing-resource and the budget for the module testing as $W_P = 1.0 \times 10^6$ and $C_P = 1.0 \times 10^6$, respectively.

Tables 2 and 3 show the estimated optimal testing-resource allocation for each module, total amount of optimal testing-resource, and utility along with the several weight patterns for the 2 and 3 attributes, respectively. In Table 2, the optimal total amount of testing-resource increases as the δ_R is increasing and the δ_E is decreasing. And we can say that the utility is ordered as P1 > P5 > P2 > P4 > P3, i.e., the utility is getting higher as the difference between δ_R and δ_E is increased. In Table 3, the optimal total amount of testing-resource is increased as δ_R is increasing and the δ_R and δ_E are decreasing. The order on the utility is P6 > P1 > P5 > P2 > P3 > P4. From Table 3, we can see the differences among the weights must influence the

Table 1 Estimated expected number of remaining faults (2 attributes)

M	a_i	r_i	z_i				
			P1	P2	P3	P4	P5
1	63	5.332×10^{-5}	21.183	5.4920	2.3573	1.0087	0.2615
2	13	2.523×10^{-4}	4.4768	1.1607	0.4974	0.2132	0.0552
3	6	5.262×10^{-4}	2.1465	0.5565	0.2385	0.1022	0.0265
4	51	5.169×10^{-5}	21.851	5.6652	2.4279	1.4050	0.2698
5	15	1.707×10^{-4}	6.6169	1.7155	0.7352	0.3151	0.0817
6	39	5.723×10^{-5}	19.736	5.1168	2.1929	0.9398	0.2437
7	21	9.938×10^{-5}	11.365	2.9466	1.2628	0.5412	0.1403
8	9	1.743×10^{-4}	6.4802	1.6801	0.7200	0.3086	0.0800
9	23	5.057×10^{-5}	22.335	5.7906	2.4817	1.0636	0.2757
10	11	8.782×10^{-5}	11.000	3.3345	1.4291	0.6125	0.1588
N^*			127.19	33.458	14.339	6.1454	1.5933

Table 2 Estimated optimal testing-resource allocation (2 attributes)

	Weight		d_i^*					
	δ_R	δ_E	M1	M2	M3	M4	M5	M6
P1	0.1	0.9	20441	4225.3	1953.5	16397	4794.5	11901
P2	0.3	0.7	45759	9575.8	4518.9	42513	12703	35489
P3	0.5	0.5	61649	12934	6129.1	58905	17666	50294
P4	0.7	0.3	77540	16292	7739.3	75297	22630	65099
P5	0.9	0.1	102858	21643	10305	101412	30538	88687
	Weight		d_i^*					Utility
	δ_R	δ_E	M7	M8	M9	M10	W^*	
P1	0.1	0.9	6177.7	1884.5	579.84	0	68355	1.0428
P2	0.3	0.7	19761	9629.4	27274	13591	220814	0.7636
P3	0.5	0.5	28287	14491	44029	23239	317624	0.7015
P4	0.7	0.3	36813	19352	60784	32888	414434	0.7585
P5	0.9	0.1	50396	27097	87478	48259	568673	0.8793

Table 3 Estimated optimal testing-resource allocation (3 attributes)

	Weight			d_i^*					
	δ_R	δ_E	δC	M1	M2	M3	M4	M5	M6
P1	0.8	0.1	0.1	70464	14797	7022.3	67998	20420	58507
P2	0.6	0.2	0.2	52070	10910	5158.4	49023	14674	41369
P3	0.5	0.25	0.25	44466	9302.5	4387.9	41179	12299	34284
P4	0.4	0.3	0.3	36862	7695.5	3617.4	33335	9923.7	27200
P5	0.2	0.4	0.4	18469	3808.5	1753.6	14363	4178.5	10064
P6	0.1	0.45	0.45	3264.8	595.32	212.98	0	0	0
	Weight			d_i^*					Utility
	δ_R	δ_E	δ_C	M7	M8	M9	M10	W^*	
P1	0.8	0.1	0.1	33017	17187	53323	28591	371326	0.4430
P2	0.6	0.2	0.2	23147	11560	33928	17423	259262	0.1661
P3	0.5	0.25	0.25	19068	9233.9	25911	12806	212936	0.1024
P4	0.4	0.3	0.3	14988	6907.7	17893	8189.5	166612	0.0966
P5	0.2	0.4	0.4	5119.6	1281.2	0	0	59036.4	0.3537
P6	0.1	0.45	0.45	0	0	0	0	4073.06	0.7266

Table 4 Estimated expected number of remaining faults (3 attributes)

	z_i					
M	P1	P2	P3	P4	P5	P6
1	1.4711	3.9227	5.8840	8.8258	23.532	52.535
2	0.3109	0.8290	1.2435	1.8652	4.9732	11.817
3	0.1491	0.3975	0.5962	0.8943	2.3846	5.3639
4	1.5174	4.0464	6.0695	9.1041	24.275	51
5	0.4595	1.2253	1.8794	2.7568	7.3506	15
6	1.3706	3.6547	5.4820	8.2228	21.925	39
7	0.7893	2.1046	3.1569	4.7353	12.626	21
8	0.4500	1.2000	1.8000	2.6999	7.1988	9
9	1.5511	4.1361	6.2040	9.3058	23	23
10	0.8931	2.3817	3.5724	5.3586	11	11
N^*	8.9620	23.898	35.846	53.768	138.26	238.49

utility. That is, we can say the testing strategy for the module testing influences on the value of the utility. Further, Tables 1 and 4 show the estimated expected number of remaining faults based on the estimated optimal testing allocation for each module for the cases of 2 and 3 attributes, respectively. Focusing on the column of P5 in Table 1, the number of remaining faults can be reduced to 1.5933 from 251 faults due to the testing-strategy, in which the software development manager set the higher

weight for the software reliability attributes. We can obtain the same investigation in Table 4.

5 Concluding Remarks

We discussed approaches for estimating optimal testing-resource allocation based on the multi-attribute utility theory in the module testing of software system. Concretely, applying the software reliability estimated the testing-effort dependent software reliability growth model, testing-resource, and testing-cost attributes for developing the additive linear form multi-attribute function, we formulated a testing-resource allocation problem, which enables us to estimate the optimal testing-resource allocation maximizing the utility of the software development manager under the certain testing-strategy. Further, we showed examples of application of our approaches by using actual data. Then, we discussed the behavior of the utility, optimal testing-resource allocation, and the number of remaining faults, which depend on the weight parameters, such as δ_R, δ_E and δ_C. From the examples of application of our approaches, we investigated that the developing the certain strategy for conducting the differentiated injections to the software reliability, testing-resource, and testing-cost leads to obtaining the higher utility of software developing manager. However, we need to conduct more investigations for our approaches and to figure out the relationship between the testing-strategy and utility by using other actual data and module testing situation.

References

Ichimori T, Tanaka M, Yamada S (2002) An optimal effort allocation problem for both module and integration testing in software development. J Jpn Ind Manag Assoc 53:201–207

Kapur PK, Singh VB, Singh O, Singh JNP (2012) Software release time based on different multi-attribute utility functions. Int J Reliab, Qual Saf Eng 20:1350012

Nishiwaki M, Yamada S, Ichimori T (1995) Testing-resource allocation policies based on an optimal software release problem. J Jpn Ind Manag Assoc 46:182–186

Pham H (2000) Software reliability. Springer, Singapore

Yamada S (2014) Software reliability modeling—fundamentals and applications. Springer Japan, Tokyo/Heidelberg

Yamada S, Ohtera H (1990) Software reliability—theory and practical application, in Japanese. Soft Research Center, Tokyo

Yamada S, Ichimori T, Nishiwaki M (1995) Optimal allocation policies for testing-resource based on a software reliability growth model. Int J Math Comput Model 22:295–301