Jestr Journal of Engineering Science and Technology Review 6 (3) (2013) 111- 118

JOURNAL OF
Engineering Science
and Technology Review

www.jestr.org

Research Article

A Study of Optimal Release Policy for SRGM with Imperfect Debugging

Ce ZHANG[1,2,*], Gang CUI[1], Fan-chao MENG[2], Hong-wei LIU[1] and Shi-xiong WU[2]

[1]*School of Computer Science and Technology, Harbin Institute of Technology, Harbin, Heilongjiang province150001 - China*
[2]*School of Computer Science and Technology, Harbin Institute of Technology at Weihai, Weihai, Shandong province, 264209 - China*

Received 5 September 2013; Accepted 25 November 2013

Abstract

In allusion to the flaws in software cost model and optimal release policy, inadequate consideration for real debugging, a cost model and optimal release policy for SRGM (*Software Reliability Growth Model*) incorporating imperfect debugging is proposed. A SRGM is presented, based on incomplete debugging, introduction of new faults and TE (*Testing Effort*). It is verified to describe real testing process well by actual failure data set and has better performance as compared to other models. Based on the proposed SRGM, a formulation of cost function is also established especially considering the impact of imperfect debugging on cost. Furthermore, the optimal release policies given limited reliability objective and the uncertainty in actual total cost exceeding expected one are developed and elaborated. Finally, a numerical example and related data analyses are illustrated and parameter sensitivity analysis is performed.

Keywords: Software Reliability, Software Cost, Optimal Release Policy, Imperfect Debugging, Testing Effort

1. Introduction

To improve reliability, testing resources are expended to detect and remove faults during the testing phase of a software product[1], [2]. Generally speaking, achieving higher reliability needs more cost and software release time could be postpone; less testing time could not incur cost overrun, but the reliability of software can't be guaranteed completely[3]. Currently, many SRGMs (*Software Reliability Growth Models*) have been proposed[4] to measure, predict and ensure reliability. Moreover, the growth of reliability, and the tradeoff between cost expenditure and optimal release both depend on the accuracy of SRGM established. So, in SRE (*Software Reliability Engineering*), building accurate SRGM, implementing effectively balance control between software cost—reliability and optimal release time are important guarantees for realizing projected objectives.

So far, many research about CM&ORPs (*Cost Model & Optimal Release Policies*) have been presented and studied[5], [6], [7], [8], [9], [10], [11], [12], [13]. They mainly focus on several areas including SRGM with perfect or imperfect debugging[8], [9], [10], [11], [12], TE (*Testing Effort*) [14], [15] and cost structure in testing and operational phase[5], [6], [7], [16]. Compared with perfect debugging, ID (*Imperfect Debugging*), a further description of software testing process, can depict more details in testing. In CM&ORPs based on imperfect debugging, some progress has been achieved[8], [9], [10], [11], [12]. Hoang Pham[8] is the first person who introduced ID from the perspective of introducing new faults into CM&ORPs, where he combined ID, penalty cost and random life cycle to serve as the foundation for optimal software release. Considering incomplete debugging, literature[9] took into account warranty cost[5], [7], [16](including the cost in operational phase) to determine optimal release policy. Moreover, based on classical J-M model, Philip J. Boland[11] also studied SRGM and CM&ORPs only with incomplete debugging. Later, P. K. Kapur[10]explicitly pointed out minimum cost ($C(T)$) can be solved by mathematical resolution, but fault removal probability (p) should be estimated by real failure data not by numerical solutions in [12], because p obtained by numerical solutions is not very likely consistent with the real testing process. Thus, Kapur presented SRGM with ID covering fault removal probability and introduction of new faults and studied CM&ORP-related problem, obtaining the good effects. The main shortcomings of the above research works are lack of deepness and of consideration of TE. On the basis of related works in TE, Huang[14], [15] incorporated generalized Logistic TEF into SRGM, established CM&ORP models considering the improvement of testing efficiency, and finally developed a more comprehensive optimal software release policy. The disadvantage lies in the lack of study of CM&ORP in ID.

In response to the problems and deficiencies of current research, the paper studies CM&ORP with ID considering TE. SGIDM (*Semi-Generalized Imperfect Debugging Model*) is proposed, software cost model across the life cycle is built, and ORP-BEVRA (*Optimum Release Policy Based on Expected Value and Risk Analysis*) is illustrated in this paper. Finally, a numerical example is verified and analyzed based on real testing data set.

2. SRGM with ID

To accurately describe real software test, the proposed SRGM is based on the following assumptions[1], [10], [12], [14], [15].

* E-mail address: zhangce@hitwh.edu.cn

(1) Fault removal process follows a NHPP (*Non-Homogeneous Poisson Process*), where $N(t)$ is the cumulative number of failures with mean value function $m(t)=E(N(t))$ and $N(0)=0$;
(2) In $(t,t+\Delta t]$, a failure occurs at most and FDR (*Fault Detection Rate*) is proportional to the remaining faults in software by the current TE expenditures;
(3) In $(t,t+\Delta t]$, fault is removed with probability $p(t)$ and new faults can be introduced with introduction rate $r(t)$;
(4) TE is modeled by improved Logistic TEF formulation.

The following differential equations can be derived from above assumptions (1)~(4):

$$\begin{cases} \dfrac{dm(t)}{dt}=\dfrac{dW(t)}{dt}\times b(t)\times\big(a(t)-c(t)\big) \\ \dfrac{da(t)}{dt}=r(t)\times\dfrac{dc(t)}{dt} \\ \dfrac{dc(t)}{dt}=p(t)\times\dfrac{dm(t)}{dt} \end{cases}$$

where $a(t)$ represents the total number of faults in software, $c(t)$ is the cumulative number of faults removed in $[0,t]$, $b(t)$ is FDR with current TE expenditures and $w(t)$ denotes testing resource consumption rate at t, that is $dW(t)/dt=w(t)$. The second formula in (1) means that the number of new faults introduced is proportional to that of the removed at t with proportion function $r(t)$. Generally speaking, fault introduction occurs in correcting not in detecting, so $da(t)/dt$ should be proportional to $dc(t)/dt$ not $dm(t)/dt$. Solving the above differential equations with the boundary condition of $m(0)=0$, $c(0)=0$, $a(0)=a$ yields:

$$m(t)=a\int_0^t b(v)w(v)\times \left[1-\int_0^v b(u)p(u)\big(1-r(u)\big)w(u)e^{\left[-\int_0^u b(\tau)p(\tau)(1-r(\tau))w(\tau)d\tau\right]}du\right]dv$$

$$a(t)=a\left[1+\int_0^t b(u)p(u)r(u)w(u)e^{\left[-\int_0^u b(\tau)p(\tau)(1-r(\tau))w(\tau)d\tau\right]}du\right]$$

$$\lambda(t)=\frac{dm(t)}{dt}=ab(t)w(t)\times \left[1-\int_0^t b(u)p(u)\big(1-r(u)\big)w(u)e^{\left[-\int_0^u b(\tau)p(\tau)(1-r(\tau))w(\tau)d\tau\right]}du\right]$$

During a testing process, with the expenditures of TE, the faults in software are continually detected and removed and the remaining faults are decreasing, which makes the growing reliability. And in lots of situations, TE consumption rate first shows an increase, and then a decrease trend. So, we present:

$$w(t)=N\left(\frac{(u+v)\alpha e^{-\alpha t}}{\left(1+ue^{-\alpha t}\right)^2}\right)$$

where N is the totally available testing resources, α is consumption rate and u and v represent the adjustment coefficient. It's easy to prove that $w(t)$ obtains maximum when $t_{\max}=\ln u/\alpha$. So, $w(t)$ shows *S*-shaped trend. Considering $dW(t)/dt=w(t)$ and the existence of $W(t)$ initialization, the following equation can be derived as:

$$W(t)=N\left(\frac{1-ve^{-\alpha t}}{1+ue^{-\alpha t}}\right)$$

This is a kind of improved Logistic TEF. At the beginning of testing ($t=0$), $W(0)=N(1-v)/(1+u)$, so $W(0)\neq 0$ which is different from that of Weibull TEF[17] and Logistic TEF[14], [15]. This indicates that testing preparation works will consume a part of testing resources at the beginning.

Hereon, The main focus centers on the effect of $W(t)$ on SRGM. Without loss of generality, let $b(t)=b$, $r(t)=r$, $p(t)=p$, thus $m(t)$ can be rewritten as:

$$m(t)=\frac{a}{p(1-r)}\left[1-e^{\left[-p(1-r)b\int_0^t w(x)dx\right]}\right]=\frac{a}{p(1-r)}\left[1-e^{\left[-bp(1-r)W^*(t)\right]}\right]$$

where $W^*(t)=\int_0^t w(x)dx=W(t)-W(0)$. It is easy to know, $m(t)$ is increasing function with testing time t. At this point, $\lambda(t)=abw(t)e^{\left[-bp(1-r)W^*(t)\right]}$ is discussed as follows: (1) $\lambda(0)=abw(0)=\dfrac{ab(u+v)\alpha}{(1+u)^2}N>0$; (2) $\lambda(\infty)=0$. Thus it can be seen that, as a whole, $\lambda(t)$ remains a decreasing trend. (3) Differentiate $\lambda(t)$ with respect to t:

$$\frac{d\lambda(t)}{dt}=\left[\frac{ab(u+v)\alpha^2Ne^{(\alpha t-p(1-r)bW^*(t))}}{(u+e^{\alpha t})^4}\right]\left[-\left(e^{\alpha t}+\frac{h}{2}\right)^2+\left(\frac{h^2}{4}+u^2\right)\right]$$

where $h=bp(1-r)(u+v)N$. Obviously, $\dfrac{d\lambda(t)}{dt}$ is decreasing function, and when $t\in\left[0,\ln\left(\left(-h+\sqrt{h^2+4u^2}\right)/2\right)/\alpha\right]$, $\dfrac{d\lambda(t)}{dt}\geq 0$; when $t\in\left(\ln\left(\left(-h+\sqrt{h^2+4u^2}\right)/2\right)/\alpha,\infty\right)$, $\dfrac{d\lambda(t)}{dt}<0$. Thus, the fact that $\lambda(t)$ shows a decreasing then increasing *S*-shaped variation trend versus time t, as indicated in most testing process[1], [18], [19], means that the established SRGM can be used to describe the testing process. Similarly, $a(t)$ can be solved as follows:

$$a(t)=\frac{a}{(1-r)}\left[1-re^{\left[-bp(1-r)W^*(t)\right]}\right]$$

Obviously, $a(t)$ is also increasing function with t. This can be explained that, due to the new faults introduction rate r, there is a rising process in the total number of faults in software. When $t\to\infty$, we can get:

$$m(t\to\infty)\approx m(\infty)=\frac{a}{p(1-r)}\left[1-e^{\left[-bp(1-r)N\left(\frac{u+v}{1+u}\right)\right]}\right]$$

$$a(t\to\infty)\approx a(\infty)=\frac{a}{(1-r)}\left[1-re^{\left[-bp(1-r)N\left(\frac{u+v}{1+u}\right)\right]}\right]$$

During testing, in time interval $(t,t+x)$, software reliability can be represented as(assume the latest failure time is t, $t\geq 0$, $x>0$):

$$R(x\mid t)=e^{-\left(m(t+x)-m(t)\right)}=e^{-\left[\frac{a}{p(1-r)}\left(\frac{e^{-bp(1-r)W(t+x)}-e^{-bp(1-r)W(t)}}{e^{-bp(1-r)W(0)}}\right)\right]}$$

Considering incomplete debugging and introducing new faults[20], the proposed SRGM in this study incorporates the

measurement of TE into modeling, so it is defined as SGIDM (*Semi-Generalized Imperfect Debugging Model*).

3. Software cost model and optimal release time policy considering ID

3.1 Cost mode considering ID

Obviously, software cost is different under perfect or imperfect debugging circumstance. Through analysis in section 2, with imperfect debugging, $m(t)$ can be influenced by p, r and $W^*(t)$ and software cost can also be influenced accordingly. Let T and T_{OPT} be software release time and optimal release time respectively, so software cost function $E[C(T, p, r)]$ can be denoted as:

$$\begin{aligned} E[C(T,p,r)] &= C_{Test}^{t<T} + C_{Operation}^{t\geq T} \\ &= \left[C_0 + C_1 m(T) + C(p,r)\int_0^T w(\tau)d\tau\right] + \\ &\quad \left[C_2\left(m(T_{LC}) - m(T)\right) + C_4\left(1 - R(x\,|\,T)\right)\right] \\ &= C_0 + C_1 m(T) + C_2\left(m(T_{LC}) - m(T)\right) + \\ &\quad C(p,r)\int_0^T w(\tau)d\tau + C_4\left(1 - R(x\,|\,T)\right) \end{aligned}$$

Software cost structure is described in Figure 1 from testing preparation (t=0) to the end of software life cycle (t=T_{LC}).

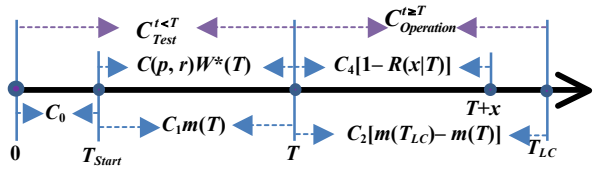

Fig. 1. Software life cycle and cost structure

As can be seen from Fig.1, cost function $E[C(T, p, r)]$ consists of testing phase cost $C_{Test}^{t<T}$ and operation phase cost $C_{Operation}^{t\geq T}$. With $W(0)\neq 0$, there exists testing cost initialization: $C_0=W(0)=N(1-v)/(1+u)$, where the related parameters can be estimated by real failure data set. C_1 and C_2 represent the expected cost of a fault removing during the testing phase and the operation phase respectively and $C_2>C_1$[3], [8], [12], T_{LC} is the length of software life cycle, and C_4 is risk cost (that is the cost incurred by software failure in operation phase). $R(x|T)$ as the reliability after releasing software, in particular, instead should be:

$$R(x\,|\,T) = e^{-\lambda(T)x} = e^{-\left[abw(T)e^{\left[-bp(1-r)W^*(T)\right]}\right]x}$$

$C(p, r)$ is testing cost by current TE expenditure during testing process including the cost caused by imperfect debugging. Hereon, $C(p, r)$ is defined as:

$$C(p,r) = \frac{C_3}{\sqrt[k]{\left[1-p(1-r)\right]}}$$

where k is imperfect debugging adjustment factor for describing the impact of imperfect debugging on cost. Min Xie[12] has introduced the concept of testing level (TL) denoted by p describing the extent of perfect debugging. And on this basis we redefine TL including fault introduction rate r as follows: $TL=p(1-r)$. Obviously, larger values of TL indicate higher perfect debugging content (namely, TL is more close to 1). In the meantime, when $p\to 1$ and $r\to 0$, $TL\to 1$ and $C(p,r)\to\infty$, which matches what one expects from perfect debugging process.

3.2 Cost mode and optimal release policy

Considering minimum of reliability R_0 that should be reached when releasing software, the optimal release problem can be formulated as the follows:

$$\begin{cases} \text{Min } E\left[C(T,p,r)\right], \\ \text{S. t. } R(x\,|\,T) \geq R_0, \textit{ where } 0 < R_0 < 1 \textit{ and } x > 0, \\ \quad \textit{for } C_2 > C_1 > 0, 0 < r << p < 1 \textit{ and } T \geq 0. \end{cases}$$

In equation above, objective function and constraints are both nonlinear with respect to argument T, so the solving process can be viewed as NPP (*Nonlinear Programming Problem*). Differentiating $E[C(T, p, r)]$ with respect to T, we can get:

$$\begin{aligned} &\frac{dE[C(T,p,r)]}{dT} = \left(C_1 - C_2\right)\frac{dm(T)}{dT} + \frac{C_3}{\sqrt[k]{\left[1-p(1-r)\right]}}w(T) - C_4\frac{dR(x\,|\,T)}{dT} \\ &= \left(C_1 - C_2\right)\lambda(T) + \frac{C_3}{\sqrt[k]{\left[1-p(1-r)\right]}}w(T) + C_4 x e^{-\lambda(T)x}\frac{d\lambda(T)}{dT} \\ &= \left[\frac{C_3}{\sqrt[k]{\left[1-p(1-r)\right]}} - \left(C_2 - C_1\right)abe^{-bp(1-r)W^*(T)}\right]w(T) + C_4 x e^{-\lambda(T)x}\frac{d\lambda(T)}{dT} \\ &= F(T,p,r) \end{aligned}$$

Substituting $\lambda(T)$ and $d\lambda(T)/dT$ into the above equation can get:

$$\begin{aligned} F(T,p,r) = &\left[\frac{C_3}{\sqrt[k]{\left[1-p(1-r)\right]}} - \left(C_2 - C_1\right)abe^{-lW^*(T)}\right]w(T) + \\ &C_4 abx\left[\frac{dw(T)}{dT} - lw^2(T)\right]e^{-\left[lW^*(T) + abxw(T)e^{-lW^*(T)}\right]}, \end{aligned}$$

where $l=bp(1-r)$. $F(T, p, r)$ is the function that depends on $w(T)$ and many other parameters. It is clear that, when $T\to\infty$, $w(\infty)=0$ and $F(\infty,p,r)=0$; when T=0,

$$\begin{aligned} F(0,p,r) = &\left[\frac{C_3}{\sqrt[k]{\left[1-p(1-r)\right]}} - (C_2 - C_1)ab\right]w(0) + \\ &\left[\frac{ab(u+v)\alpha^2 N\left(u^2 - 1 - h\right)C_4 x}{(1+u)^4}\right]e^{-abw(0)x}. \end{aligned}$$

Obviously, due to the complexity of parameters, there exists possibility of $F(0, p, r)>0$ or $F(0, p, r)<0$. In summary, $\left|\frac{dE[C(T,p,r)]}{dT}\right|$ remains downward trend, and is gradually tending to 0. Due to the uncertainty in relation of $\left|\frac{dE[C(T,p,r)]}{dT}\right|$ and 0, variation tendency (increasing function or decreasing function) of $E[C(T, p, r)]$ is not directly determined by $F(T, p, r)$. So, nonlinear and complex $F(T, p, r)$ means the above optimization problem has to be solved by numerical calculation.

Furthermore, through the equation of $R(x|T)$, $R(x|T)$ is a increasing function, thus we can get: if $R(x|0)<R_0$, then exists $T\geq T_{R(x|T)\geq R0}= T_1$ that makes $R(x|T)\geq R_0$ met, that is $R(x|T_1)=R_0$; if $R(x|0)\geq R_0$, then for $\forall T\geq 0$, $R(x|T)\geq R_0$, that is $T_{R(x|T)\geq R0}= T_1=0$.

3.3 Risk analysis of cost mode

Obviously, based on the above analysis, the obtained minimum cost is *ETC* (*Expected Testing Cost*). Bo Yang has noted that[21] *ATC* (*Actual Testing Cost*): $C(T, p, r)$ is a random variable and it may have a serious deviation from *ETC*: $E[C(T, p, r)]$. There is a risk that *ATC* may also be larger than *ETC* (a certain extent uncertainty), resulting in the serious problem of budget overruns during software testing. So, it is necessary to conduct a risk evaluation. Thus, the optimal release problem can be extended as:

$$\begin{cases} \text{Min } E\left[C(T,p,r)\right], \\ \text{S. t. } R(x \mid T) \geq R_0, \text{ where } 0 < R_0 < 1 \text{ and } x > 0, \\ \quad P_\sigma(T) < \delta, \text{ where } P_\sigma(T) = \Pr\left\{C(T,p,r) > (1+\sigma)E\left[C(T,p,r)\right]\right\} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad , 0 < \sigma, \delta < 1, \\ \quad \text{for } C_2 > C_1 > 0, C_3 > 0, C_4 > 0, 0 < r << p < 1 \text{ and } T \geq 0. \end{cases}$$

The second condition indicates $P_\sigma(T)$ is a risk function, the probability of *ATC* exceeding $(1+\sigma)E\left[C(T,p,r)\right]$. By setting coefficient $(1+\sigma)$, *ATC* exceeding *ETC* is limited to a specified range: threshold δ. Substituting $E[C(T,p,r)]$ into the equation and by calculating $P_\sigma(T)$, we can get:

$$P_\sigma(T) = 1 - e^{-m(T_{LC})} \sum_{i=0}^{I} \sum_{j=0}^{\left\lfloor \frac{C_4}{C_2}(\chi - i) \right\rfloor} \left[\frac{(m(T))^i}{i!} \right] \left[\frac{(m(T_{LC}) - m(T))^j}{j!} \right]$$

where

$$I = \begin{bmatrix} \chi = \dfrac{\sigma}{C_1}\left[C_0 + C(p,r)W^*(T) + C_4\left(1 - R(x \mid T)\right)\right] \\ + \dfrac{(1+\sigma)}{C_1}\left[C_2\left(m(T_{LC})\right) - \left(C_2 - C_1\right)m(T)\right] \end{bmatrix}$$

and $\lfloor x \rfloor$ denotes the largest integer value that is less than or equal to x.

3.4 Optimal release time algorithm: *ORP-BEVRA*

On the basis of the above comprehensive analysis, the solving algorithm: *ORP-BEVRA* (***O**ptimum **R**elease **P**olicy-**B**ased on **E**xpected **V**alue and **R**isk **A**nalysis*) about existence and the uniqueness for the optimum solution is presented.

1:	***ORP-BEVRA***(**O**ptimum **R**elease **P**olicy-**B**ased on **E**xpected **V**alue and **R**isk **A**nalysis)
2:	**If** $F(0, p, r)<0$ **and** $F(T_{LC}, p, r)>0$, **then** exist a unique solution T_0 makes $F(T_0, p, r)=0$.
3:	In this case **if** $R(x\|0)<R_0$, **then** $T^*=\max(T_0, T_1)$;
4:	**If** $F(0, p, r)<0$ **and** $F(T_{LC}, p, r)>0$, **then** exist a unique solution T_0 makes $F(T_0, p, r)=0$.
5:	In this case **if** $R(x\|0) \geq R_0$, **then** $T^*=\max(T_0, 0)$;
6:	**If** $F(0, p, r)<0$ **and** $F(T_{LC}, p, r)<0$ **and** $R(x\|0)<R_0$, **then** $T^*=\max(T_{LC}, T_1)$;
7:	**If** $F(0, p, r)<0$ **and** $F(T_{LC}, p, r)<0$ **and** $R(x\|0) \geq R_0$, **then** $T^*=T_{LC}$;
8:	**If** $F(0, p, r)>0$ **and** $F(T_{LC}, p, r)>0$ **and** $R(x\|0)<R_0$, **then** $T^*=T_1$;
9:	**If** $F(0, p, r)>0$ **and** $F(T_{LC}, p, r)>0$ **and** $R(x\|0) \geq R_0$, **then** $T^*=0$;
// T^* is quasi-optimal release time and can be calculated and obtained by the combination of $F(T, p, r)$ curve shape and reliability constraint condition ($R(x\|T) \geq R_0$).	
10:	**If** $P_\sigma(T^*) < \delta$ **then** $T_{OPT} = T^*$;
11:	**Else** obtain the solution of $P_\sigma(T) < \delta$, **and** denote it with T_{Risk} ($T_{Risk} > T^*$), **then** $T_{OPT} = T_{Risk}$;

In real situation, based on *ORP-BEVRA*, firstly, quasi-optimal release time T^* can be obtained, then a risk evaluation is conducted according to risk objective δ set by software project manager，and finally optimal release time T_{OPT} can be obtained.

4. Numerical examples

In this section, to demonstrate the validity of proposed model, we apply the proposed SRGM, cost model and optimal release policy to a real software failure data set.

4.1 SRGMs performance validation

4.1.1 Comparison criteria and data set

In order to verify the feasibility of the models, we selects failure data set [22] that has been widely used to illustrate the performance of SRGMs. In the meanwhile, to quantitatively differentiate the differences in the models, the following fitness and prediction criteria are adopted:

$$MSE = \frac{1}{K}\sum_{i=1}^{K}\left(m(t_i) - n_i\right)^2$$

$$MEOP = \sum_{i=k}^{K}\left|m(t_i) - n_i\right| \Big/ (n-k+1)$$

$$Variation = \sqrt{\sum_{i=1}^{K}\left(\left(n_i - m(t_i)\right) - Bias\right)^2 \Big/ (K-1)}$$

$$RMSPE = \sqrt{Variation^2 + Bias^2}$$

$$Bias = \frac{1}{K}\sum_{i=1}^{K}\left(n_i - m(t_i)\right)$$

$$R-square = \frac{\sum_{i=1}^{K}\left[m(t_i) - \bar{n}\right]^2}{\sum_{i=1}^{K}\left[n_i - \bar{n}\right]^2}, \bar{n} = \frac{1}{K}\sum_{i=1}^{K} n_i$$

$$TS = \sqrt{\sum_{i=1}^{K}\left(m(t_i) - n_i\right)^2 \Big/ \sum_{i=1}^{K} n_i^2} \times 100\%$$

$$RE = \frac{m(t_q) - q}{t_q}$$

where K is the number of failure data sample, n_i and $m(t_i)$ respectively represent the cumulative number of failures and estimated value of faults by t_i. Smaller values of *MSE*, *MEOP*, *Variation*, *RMSPE* and *TS*, and the closer to 1 of *R-square* indicate better fitting accuracy. The quickly closer to 0 of *RE* reveals better prediction performance.

4.1.2 Model performance analysis

The comparisons between the observed data and the proposed model are illustrated graphically in Fig.2. For comparison purposes, the fitting of three typical SRGMs[10], [23], [24] related to TEF and/or ID are also shown in Fig.2. Fig.3 demonstrates the relative error (*RE*) curves for the models.

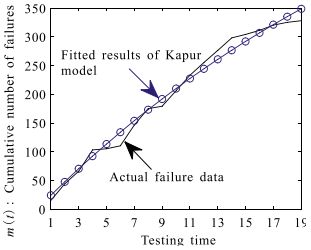

Fig. 2-1. Kapur model

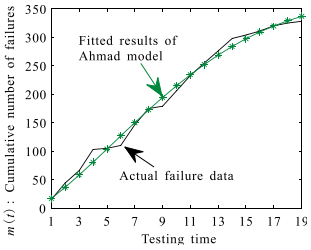

Fig. 2-2. Ahmad model

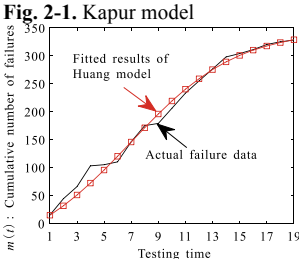

Fig. 2-3. Huang model

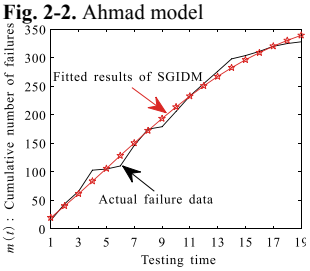

Fig. 2-4. SGIDM

Fig. 2. Comparison of goodness of fitting of the models

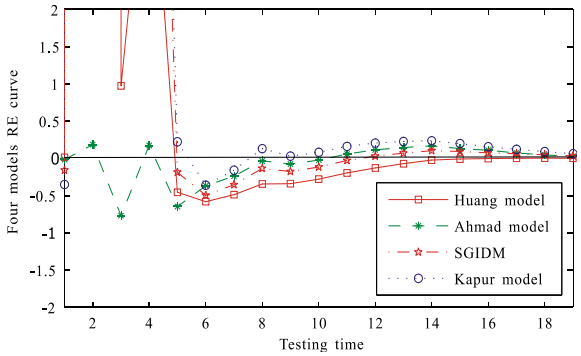

Fig. 3. *RE* curves of the models

Thus it can be seen that, SGIDM and actual failure curves are closely overlapped. In order to make a qualitative comparison, the evaluation criteria values of the models are listed in Table 1. From Table 1, we can see that the proposed model (SGIDM) has lower value of *MSE*, *Variation*, *RMSPE*, *TS*, *MEOP*, and *R-square* value is closer to 1. On average, SGIDM yields a better fit for this data set and fits significantly better than the others. On the other hand, it is clear from Fig.3 that the *RE* value of SGIDM can quickly approach zero in comparison with the other models and provide better performance of prediction.

Table 1 Comparison of descriptive power of the models

Model	Huang Model[23]	Ahmad Model[24]	Kapur Model[10]	SGIDM
MSE	114.085 11020	85.9633 8226	139.815	84.8733 3754
Variation	10.7718 9151	9.50029 390	12.0892 307	9.4646 0056
RMSPE	11.7367 0504	9.62537 492	12.3790 071	9.46717 703
R-square	1.06795 470	1.01778 405	0.94130 928	0.99552 145
TS	4.7624 28%	4.1340 02%	5.2721 90%	4.1077 08%
MEOP	7.49602 081	7.07343 043	9.89065 084	7.15859 585

Consequently, from Fig.2-3 and Table1, it can be concluded that the proposed model (SGIDM) fits the real failure data better than the others and gets reasonable prediction in estimating the number of software errors.

4.2 Optimal software release time problem

Now we will explain optimal release time problem based on the cost parameters in literature[7], [14], [15] and appropriate adjustments. In particular, it's important to point out that C_0 should be calculated (C_0=$W(0)$=$N(1-v)/(1+u)$) based on parameters obtained by real failure data set.

Firstly, based on *ORP-BEVRA*, optimal release time (T_{OPT}), minimal expected cost ($E[C(T_{OPT},p,r)]$), the cumulative number of faults detected by T_{OPT} ($m(T_{OPT})$), and reliability by T_{OPT} ($R(x|T_{OPT})$) are calculated as shown in Table 2. In comparison, the cost model and optimal release algorithm related to imperfect debugging in literature[10] are also incorporated. It should be pointed out that Xie model[12] is not included here, because it has been proved that fault removal probability p is obtained by numerical method not estimated by real failure data set. Furthermore, expected cost curve of Kapur model and the proposed model (SGIDM) are depicted in Fig.4. It can be seen that two curves both show firstly an increase-and-then-a-decrease trend, so we can conclude that there exists minimum for both. Table 2 lists the values of T_{OPT}, $E[C(T_{OPT},p,r)]$, $m(T_{OPT})$ and $R(x|T_{OPT})$ of two models.

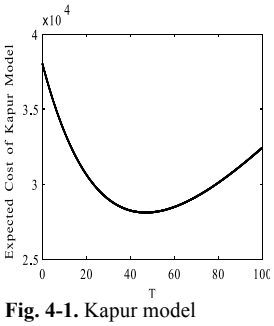

Fig. 4-1. Kapur model

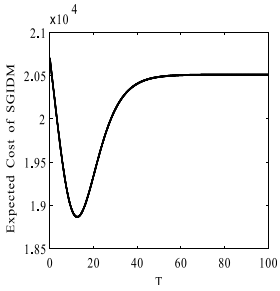

Fig. 4-2. SGIDM

Fig. 4. Expected cost curve

Table 2 Model result comparisons for (C_0,C_1,C_2,C_3,C_4) = (1.14,25,50,80,1000), x=0.05, R_0=0.85, k=2, T_{LC}=100 weeks

Model	T_{OPT}	$E[C(T_{OPT},p,r)]$	$m(T_{OPT})$	$R(x\|T_{OPT})$
Kapur model	62.65	28629.24	659.56	0.850006
SGIDM	26.19	19878.58	378.62	0.850090

As seen from Table 2, it is found that two results of experiment are significantly different: compared with Kapur model, the optimal release time of SGIDM is much shorter, its minimal expected cost much lower than that of Kapur model, and reliability obtained by optimal release time relatively higher. The reason for this is that the two SRGMs are different in nature in describing software testing process. In section 4.1, the proposed model has been proved to be optimal. By contrast, although incomplete debugging and introducing new faults are considered in Kapur model, the description and formulation are "rough" (the expression of $a(t)$ is set subjectively) and TE is not incorporated either. Furthermore, risk cost in operational stage is also ignored in Kapur model.

Next, we investigate the relationship of the size between $E[C(T, p, r)]$ and $C(T, p, r)$, and conduct uncertainty risk evaluation. Without loss of generality, let $\sigma = 0.1$, $\sigma = 0.2$, $\delta = 0.1$, $\delta = 0.15$, and then $P_\sigma(T) < \delta$ can be calculated. The curves of $P_\sigma(T)$ with $\sigma = 0.1$ and $\sigma = 0.2$ have been drawn in Fig.5. It shows that the two curves present decreasing trend and the maximum are respectively 0.02 and 3×10^{-5}. Table 3 further illustrates the relationship between $P_\sigma(T)$ and δ.

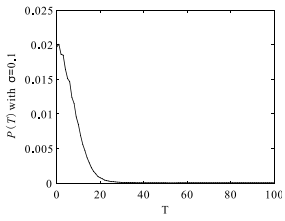

Fig. 5-1. σ= 0.1

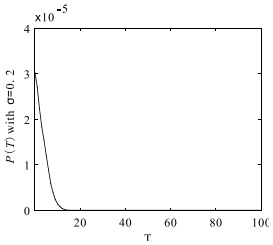

Fig. 5-2. σ= 0.2

Fig. 5. Curve of $P_\sigma(T)$

Table 3 $P_\sigma(T) < \delta$ for varies values of σ and δ

Case	$P_\sigma(T) < \delta$ ($\sigma = 0.1$, $\delta = 0.1$)	$P_\sigma(T) < \delta$ ($\sigma = 0.1$, $\delta = 0.15$)	$P_\sigma(T) < \delta$ ($\sigma = 0.2$, $\delta = 0.1$)	$P_\sigma(T) < \delta$ ($\sigma = 0.2$, $\delta = 0.15$)
$P_\sigma(T)_{\max}$	0.02		3×10^{-5}	
Result	$[P_{0.1}(T)] < [\delta = 0.1]$	$[P_{0.1}(T)] < [\delta = 0.15]$	$[P_{0.2}(T)] < [\delta = 0.1]$	$[P_{0.2}(T)] < [\delta = 0.15]$
	Yes	Yes	Yes	Yes

From Fig.5 and Table 3, we can see that the final results are $P_\sigma(T) < \delta$ that indicates that actual testing cost expenditure is approximate to expected testing cost and there is no risk of cost overrun.

Fig.6 graphically explores three dimensional image of the cumulative number of faults detected, cost and software release time.

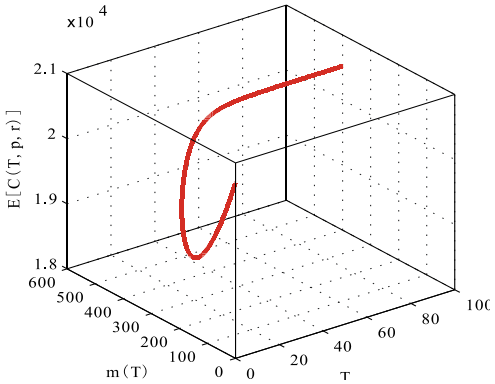

Fig. 6. The cumulative number of faults detected and cost with release time

As shown in Fig.6, taken as a whole, an extension of release time T results in the growth of cost $C(T, p, r)$. Being similar to S-shaped $\lambda(t)$ discussed in section 2, $C(T, p, r)$ also has the characteristics of decreasing firstly and then increasing and it tends to be stabilized gradually about 40 weeks later. From Fig.6, $m(T)$ continues to grow over release time T, but displays significant-initially-and-slow-later-increasing trend. It can be explained by the fact that simple faults that are easy to be detected and removed account for a relatively large proportion during initial testing, and the proportion of complex faults increases gradually as the testing proceeds. Accordingly, under the influence of imperfect debugging (mainly reflected by $m(T)$), the rising speed of $C(T, p, r)$ varies from high to low. This means that, in software testing, in order to get adequate fault removal probability(p) and introduction rate (r), to establish accurate SRGM, to manage effectively testing cost and determine optimal software release planning, available testing resources(including test cases, CPU time and man power, etc.) should be allocated reasonably according to the projected target, and testing process should be improved and optimized dramatically.

4.3 Sensitivity analysis

Due to many parameters in $E[C(T, p, r)]$, sensitivity analysis is conducted on p, r, k, C_3 and their combinations, and could be similar in other situations. Sensitivity analysis of parameter θ can be performed by the following formula:

$$S_{RC} = \frac{ROV - IOV}{IOV} = \begin{cases} \dfrac{T_{OPT}((1+v)\theta) - T_{OPT}(\theta)}{T_{OPT}(\theta)} \times 100\%, \text{single parameter} \\ \dfrac{T_{OPT}((1+v_1)\theta_1,(1+v_2)\theta_2,...,(1+v_i)\theta_i) - T_{OPT}(\theta_1,\theta_2,...\theta_i)}{T_{OPT}(\theta_1,\theta_2,...\theta_i)} \times 100\%, \text{multiple parameter} \end{cases}$$

where IOV is initial optimization value, ROV is regained optimization value and v_i is relative changes of θ_i. Hereon, relative changes of parameters are set ±10%、±20% and ±30%. Optimal release time (T_{OPT}), expected cost ($E[C(T_{OPT},p,r)]$) and reliability by T_{OPT} ($R(x|T_{OPT})$) are considered when the parameters of $E[C(T, p, r)]$ change.

4.3.1 Single parameter sensitivity analysis

Fig.7 plots sensitivity analysis results of p, r, k and C_3, including T_{OPT}, $E[C(T_{OPT},p,r)]$ and $R(x|T_{OPT})$. From Fig.7-1, smaller value of p affects optimal release time significantly, and larger value of p affects expected cost largely. For example, when p decreases from 30% to 10%, optimal release time (T_{OPT}) increases from 5% to 2.8 expanding nearly 3.62 times, and meanwhile, reliability by T_{OPT} improves by a factor of 18% over delay of optimal release time. It can be explained that, this decrease of p causes a decline in complete removal probability, and thus repairing the same amount of faults takes more time, which results in delay in optimal release time. On the other hand, when p increases from 10% to 30%, expected cost ($E[C(T_{OPT},p,r)]$) increases from 5.5% to 68% and expands nearly 1.6 times. In this course, optimal release time precedes baseline values from 4% to 11%. By examining Fig.7-2, positive and negative changes of r have some influences on T_{OPT} and $E[C(T_{OPT},p,r)]$, but these influences are not significant (<1%).

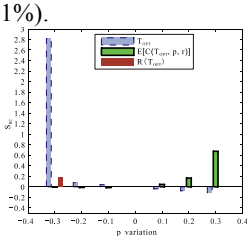

Fig. 7-1. Sensitivity analysis: p

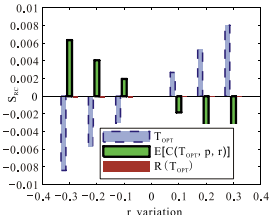

Fig. 7-2. Sensitivity analysis: r

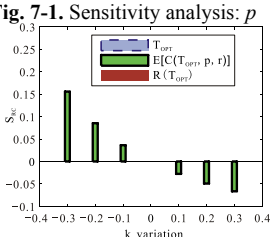

Fig. 7-3. Sensitivity analysis: k

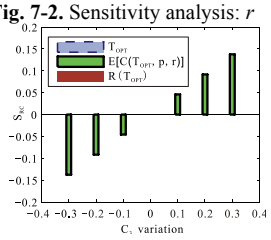

Fig. 7-4. Sensitivity analysis: C_3

Fig. 7. Single parameter sensitivity analysis

From Fig.7-3 and Fig.7-4, k and C_3 have some influence on $E[C(T_{OPT},p,r)]$, but no influence on T_{OPT} and $R(x|T_{OPT})$ (this is not reflected in Fig.7-3 and Fig.7-4). For example, when C_3 decreases from 10% to30% and increases from 10% to 30% respectively, $E[C(T_{OPT},p,r)]$ decreases from

4.5% to 13.8% and increases from 4.7% to 13.8%. So, it can be concluded that the changes of C_3 have great effect on the $E[C(T_{OPT},p,r)]$. Sensitivity analysis of the other single parameters can also be conducted in a similar fashion.

4.3.2 Multiple parameter sensitivity analysis

Fig.8 shows the results of partial two-parameter sensitivity analysis. From Fig.8-1, during the decreasing of p and C_3 from 10% to 30%, T_{OPT} postpones from 4% to 2.8 (expanding nearly 3.65 times), $E[C(T_{OPT},p,r)]$ decreases from 5% to 12%, and $R(x|T_{OPT})$ improves 18%. The reason is that, when p decreases, the curve shape of expected cost changes from decreasing-then-increasing to decreasing, which postpones optimal release time. Thus, we can see that negative changes of p and C_3 have significant influences on T_{OPT} and some influence on $R(x|T_{OPT})$; positive changes of p and C_3 have large influence on $E[C(T_{OPT},p,r)]$, some influence on T_{OPT} and no influence on $R(x|T_{OPT})$. Finally Fig.7 and Fig.8-1 reveal that p has significant influence on T_{OPT} and C_3 has some influence on $E[C(T_{OPT},p,r)]$. Similarly, the changes of p and k have influences on T_{OPT} and $E[C(T_{OPT},p,r)]$: when p and k decrease 30%, T_{OPT} is delayed about 13.5% and $E[C(T_{OPT},p,r)]$ increases 6%; when both increase 30%, $E[C(T_{OPT},p,r)]$ increases 28%, T_{OPT} is brought forward by 11.5%, and $R(x|T_{OPT})$ remains unchanged. Finally, the positive or negative changes of k and C_3 have low influence on $E[C(T_{OPT},p,r)]$ (<5%) and no influence on T_{OPT} and $R(x|T_{OPT})$.

Similarly, sensitivity analysis of the other multiple parameter combinations can also be conducted. Altogether, changes of these parameters have subtle influence on $R(x|T_{OPT})$ and different levels of influence on $E[C(T_{OPT},p,r)]$. Especially, p, r, the combination of p and k as well as p and C_3 have influences on T_{OPT}, and the influence brought by p is more significant. With imperfect debugging close to real testing, software engineer can allocate available resources based on much experience and knowledge of the system being tested, decide the optimal parameters to model SRGM, determine cost model and optimal release policy, which boosts their testing efficiency.

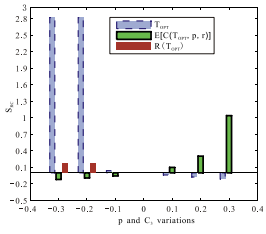

Fig. 8-1. Sensitivity analysis: p, C_3

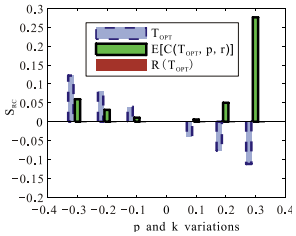

Fig. 8-2. Sensitivity analysis: p, k

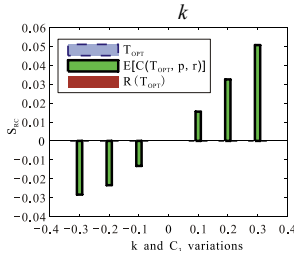

Fig. 8-3. Sensitivity analysis: k, C_3

Fig. 8. Multiple parameter sensitivity analysis

5. Conclusions

During software testing, reliability, testing resources and cost, and release time are interrelated and interdependent factors, so determining proper correlation model is the key to optimizing testing process. The paper presents SRGM with ID, cost model and optimal release policy and studies the effect of ID on SRGM, cost model and optimal release. Compared with other models, the proposed SRGM fits the failure data better and predicts the future behavior well. On this basis, ID is incorporated into software cost model and optimal release policy based on risk analysis is elaborated. In real software engineering situations, software testing is subject to imperfect debugging, so software engineer should optimize testing process making a balance between cost and reliability objective.

6. Acknowledgments

This research was supported in part by the National Key R&D Program of China(No.2013BA17F02), the National Nature Science Foundation of China (No.60503015) and the Shandong province Science and Technology Program of China(No.2011GGX10108, 2010GGX10104).

References

1. Lin, C. T., Huang, C. Y., "Enhancing and measuring the predictive capabilities of testing-effort dependent software reliability models", The Journal of Systems and Software 81, 2008, pp. 1025-1038.
2. Jha, P. C., Gupta, D., Yang, B., et al., "Optimal testing resource allocation during module testing considering cost, testing effort and reliability", Computers & Industrial Engineering 57(3), 2009, pp. 1122-1130.
3. Boehm, B., Abts, C., Chulani, S., "Software development cost estimation approaches-A survey", Annals of Software Engineering 10(1-4), 2000, 177-205.
4. Sharma, K., Garg, R., Nagpal, C. K., et al., "Selection of optimal software reliability growth models using a distance based approach", IEEE Transactions on Reliability 59(2), 2010, pp. 266-276.
5. Pham, H., Zhang, X., "A software cost model with warranty and risk costs", IEEE Transactions on Computers 48(1), 1999, pp. 71-75.
6. Zhang, X., Pham, H., "A software cost model with error removal times and risk costs", International Journal of Systems Science 29(4), 1998, pp. 435-442.
7. Zhang, X., Pham, H., "A software cost model with warranty cost, error removal times and risk costs", IIE transactions 30(12), 1998, pp. 1135-1142.
8. Pham, H., "A software cost model with imperfect debugging, random life cycle and penalty cost", International Journal of Systems Science 27(5), 1996, pp. 455-463.
9. Williams, D., "Study of the warranty cost model for software reliability with an imperfect debugging phenomenon", Turk J Elec Engin 15(3), 2007, pp. 369-381.
10. Kapur, P. K., Gupta, D., Gupta, A., et al., "Effect of introduction of fault and imperfect debugging on release time", Ratio Mathematica 18 (Journal of Applied Mathematics), 2008, pp. 62-90.
11. Boland, P. J., Ní, Chuív. N., "Optimal times for software release when repair is imperfect", Statistics & probability letters 77(12), 2007, pp. 1176-1184.
12. Xie, M., Yang, B., "A study of the effect of imperfect debugging on software development cost", IEEE Transactions on Software Engineering 29(5), 2003, pp. 471-473.
13. Kapur, P. K., Pham, H., "Aggarwal A G, et al. Two Dimensional Multi-Release Software Reliability Modeling and Optimal Release Planning", IEEE Transactions on Reliability 61(3), 2012, pp. 758-768.
14. Huang, C. Y., Lyu, M. R., "Optimal release time for software systems considering cost, testing-effort, and test efficiency", IEEE Transactions on Reliability 54(4), 2005, pp. 583-591.

15. Huang, C. Y., "Cost-reliability-optimal release policy for software reliability models incorporating improvements in testing efficiency", Journal of Systems and Software 77(2), 2005, pp. 139-155.
16. WILLIAMS, D. R. P., "Optimal release policies for a software system with warranty cost and change-point phenomenon", Turkish Journal of Electrical Engineering & Computer Sciences 21, 2013, pp. 234-245.
17. Ahmad, N., Bokhari, M. U., Quadri, S. M. K., et al., "The exponentiated Weibull software reliability growth model with various testing-efforts and optimal release policy: a performance analysis", International Journal of Quality & Reliability Management 25(2), 2008, pp. 211-235.
18. Huang, C. Y., "Performance analysis of software reliability growth models with testing-effort and change-point", Journal of Systems and Software 76(2), 2005, pp. 181-194.
19. Kapur, P. K., Singh, V. B., Anand, S., et al., "Software reliability growth model with change-point and effort control using a power function of the testing time", International Journal of Production Research 46(3), 2008, pp. 771-787.
20. Zhang, C., Cui, G., Liu, H. W., et al., "Unified and Flexible SRGM Framework Incorporating Two Types of Imperfect Debugging", Journal of Convergence Information Technology 8(8), 2013, pp. 751-758.
21. Yang, B., Hu, H., Jia, L., "A study of uncertainty in software cost and its impact on optimal software release time", IEEE Transactions on Software Engineering 34(6), 2008, pp. 813-825.
22. Ohba, M., "Software reliability analysis models", IBM Journal of research and Development 28(4), 1984, pp. 428-443.
23. Huang, C. Y., Kuo, S. Y., Lyu, M. R., "An assessment of testing-effort dependent software reliability growth models", IEEE Transactions on 56(2), 2007, pp. 198-211.
24. Ahmad, N., Khan, M. G. M., Rafi, L. S., «A study of testing-effort dependent inflection S-shaped software reliability growth models with imperfect debugging", International Journal of Quality & Reliability Management 27(1), 2010, pp. 89-110.