软件测试资源与成本管控和最优发布策略

张 策[1,2], 崔 刚[1], 刘宏伟[1], 孟凡超[2], 傅忠传[1]

(1.哈尔滨工业大学 计算机科学与技术学院,150001 哈尔滨;
2.哈尔滨工业大学(威海) 计算机科学与技术学院, 264209 山东 威海)

摘 要: 为全面系统地了解软件测试过程中与可靠性相关的重要过程的研究现状和最新发展,对测试资源和成本的管理,以及最优发布技术问题进行了深入研究.首先将研究问题划分为4类,剖析了测试资源分配与控制,成本模型和最优发布基本内容,给出了其公式化的形式化描述和内在关系;然后从软件体系结构建模、测试过程建模、非线性最优化与仿真求解3个方面进行了技术分类评述.进一步将研究内容从5个方面进行归一化分类,并对选取的典型模型进行了归类比较总结.最后,指出了进一步研究发展的主要问题.研究分析表明,测试资源分配与成本管控和最优发布策略中,考虑到软件体系结构特征和不完美排错模型,向多目标融合方向发展,并与可靠性评估相关是后续研究的研究.

关键词: 测试资源;可靠性;成本;软件可靠性增长模型;最优发布

中图分类号: TP311 **文献标志码**: A **文章编号**: 0367-6234(2014)05-0051-08

Software test resources and cost control and optimal release policy

ZHANG Ce[1,2], CUI Gang[1], LIU Hongwei[1], MENG Fanchao[2], FU Zhongchuan[1]

(1. School of Computer Science and Technology, Harbin Institute of Technology, 150001 Harbin, China;
2. School of Computer Science and Technology, Harbin Institute of Technology at Weihai, 264209 Weihai, Shandong, China)

Abstract: To reveal systematically the state of art of relevant sub-processes related to reliability in software test, management of test resource and cost, optimal release are studied in this article. Firstly, research contents are divided into four parts. Secondly, test resource allocation and control, cost model, optimal release, and relation of four parts are discussed, and then the formulations are also described. Thirdly, the related developing technologies are classified and summarized in accordance with software architecture modeling, test process modeling and nonlinear optimum or simulation solving. Furthermore, research contents are unified from five aspects and the latest models are classified and summarized respectively. Finally, future research directions are put forward.

Keywords: test resource; reliability; cost; software reliability growth model; optimal release

以可靠性为核心的软件质量通常与测试花费的时间、测试资源的消耗行为、测试方法紧密相关.测试时间延长,成本上升,可靠性获得增长,但会增加推迟软件发布的风险[1];测试时间不足,成本受控,但发布后可靠性难以保证,增大了失效导致的连带风险.显然,测试过程中,测试资源的管理,确定影响成本的因素以及决策发布时间,对软件质量自身与软件公司的声誉有着重要影响.

本文对图1软件测试中以可靠性为目标的软件测试资源管控,成本模型与最优发布策略内容进行综合评述.首先介绍有关概念,将研究内容划分为4个部分进行阐释,并给出其相互关系内涵,从实现技术分类角度讨论了研究内容间差异;然后将其归一化为5个子方面并选取典型模型进行归类比较,最后给出结论.

1 基本概念

定义1 测试资源(testing resource,TR).

收稿日期: 2013-07-24.
基金项目: 国家科技支撑计划资助项目(2013BA17F02);山东省科技攻关资助项目(2011GGX10108, 2010GGX10104).
作者简介: 张 策(1978—),男,讲师,博士研究生;
崔 刚(1949—),男,教授,博士生导师;
刘宏伟(1971—),男,教授,博士生导师.
通信作者: 张 策, zhangce@hitwh.edu.cn.

TR 指软件测试过程中所花费的资源,包括人力成本、CPU 时间、执行的测试案例数量等,也称之为测试工作量(testing effort, TE)[2-4].

定义 2 软件可靠性增长模型(software reliability growth model,SRGM).

SRGM 基于失效数据,描述软件测试过程中累积检测的故障数量,TE 与测试时间等的数学关系,是实现建模软件可靠性提高过程的数学工具.

定义 3 最优发布策略(optimal release policy,ORP).

ORP 为达到测试的预期目标而确定软件测试的最优时间长度.

定义 4 软件成本(Cost).

软件成本是指软件测试与运行阶段为提高可靠性与确保软件正常运行以及不期事件发生而连带引发等所花费的成本总和.

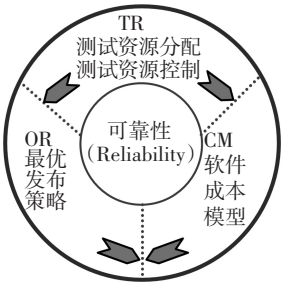

图 1 以可靠性为核心的测试资源与成本管控和最优发布

2 研究问题与分类评述

软件测试在检测与排除掉 Bug 的过程中,不断提高可靠性等质量属性.由于测试过程本质上是一个受多因素制约的随机过程,其涵盖较多的研究内容.在对现有相关研究工作进行分类和梳理的基础上,图 2 描述了在软件测试中以发布预期要求软件为主要目标的相关研究活动内容.

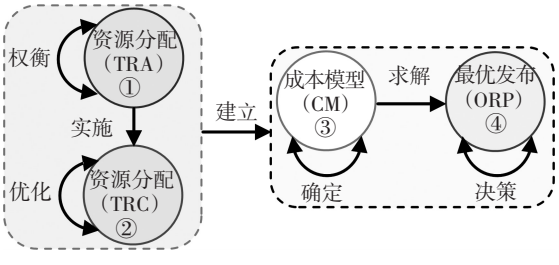

图 2 研究问题分类及内在关系

从图 2 可以看出:1)资源分配为模块/构件分配一定量的测试资源;2)资源控制对所分配资源的消耗过程进行定量约束;3)成本模型实现软件总成本的建模与控制,对风险超支进行管理;4)最优发布实现在既定目标下最优发布可信的软件制品.总体而言,测试资源管理和控制可实现资源的有效分配与消耗利用,对于达到预期可靠性以及成本不超支可建立重要保障,同时也为软件的最优发布决策建立重要决策支持.这样,也为确保发布资源消耗与可靠性受控的软件提供保障.

2.1 测试资源分配

软件测试中,软件管理者首先应对资源进行有效分配.当前,测试资源分配主要应用在构件软件(component-based software, CBS)的模块测试中[4],可分类描述为表 1 中的 5 种情况.显然,从最小化剩余故障或软件测试成本的角度来进行测试资源分配研究居多.这里,需要特别指出:

1)表 1 中的第 1 和 4 种情况并非等价,例如,在不完美排错情况下[16],软件中总的故障数量可能由于修改中引入新故障现象而存在一定量增加的过程.

2)测试时间分配在效用上与测试资源(预算)分配是可以等效的[9-10,13,17].这是因为,在上个世纪 80 年代,TE 被认为是测试时间[17],但自 90 年代开始后,测试资源被认为是 TE[4],或者是 TE 的一种表示.

3)即使已采用了最优分配策略,所排除掉的故障也是有限的,因为测试会受到时间与成本的限制.

表 1 中 5 种模块间最优的资源分配方案均是从不同的单一目标提出的优化模型,适用于单因素限制下的情况.显然,提出多因素下最优分配模型也是学者们研究的重点.另外,测试资源分配应兼顾到体系结构和构件间的差异特征,应结合更多的实际来统筹考虑.

2.2 测试资源控制

测试资源控制用以对所分配资源的消耗过程进行优化控制,提高测试效率[18-19],实现可靠性的预期增长.典型地,该问题可以描述为:1) 设 CBS, $S = \{C_i \mid 1 \leqslant i \leqslant N\}$, 规定模块测试的终止时间为 T_2;2) 基于所建立的 SRGM,C_i 的测试资源消耗行为在 $T_1(0 < T_1 < T_2)$ 被估计;3) 软件管理人员根据 T_2 时设定的目标(例如,应达到预期剩余故障) 应确定在 T_1 为 C_i 分配合适的测试资源 e_i.

另一方面,为了检测更多的故障,软件管理人员在必要的时间点进行 TE 的主动调整或可引入新的自动化工具与技术实现资源的合理控制,提高效率.此时,就需要从前后成本变化上进行更多的考虑.

事实上,测试资源控制是资源分配的一种特例,是对资源分配的动态优化过程,应始终以能够提高故障检修率和实现预定可靠性为目标.

表 1　测试资源分配策略比较

① 最小化剩余故障[3-7]	② 最小化测试资源花销[4-6, 8]	③ 最大化可靠性[9-10]	④ 最大化检测到的故障个数[11-13]	⑤ 最小化测试成本[2,6, 14-15]
$\begin{cases}\min\sum_{i=1}^{N}(a(\infty)-m(T))\\ \text{s.t.}\sum_{i=1}^{N}e_i=E, e_i\geqslant 0\end{cases}$	$\begin{cases}\min\sum_{i=1}^{N}e_i\\ \text{s.t.the remaining}\\ \text{faults is } Z\end{cases}$ or $\begin{cases}\min\sum_{i=1}^{N}e_i\\ \text{s.t.}R\geqslant R_0\end{cases}$	$\begin{cases}\max R(t)\\ \text{s.t.}\sum_{i=1}^{N}e_i\leqslant E, e_i\geqslant 0\end{cases}$ or $\begin{cases}\max R(t)\\ \text{s.t.}\sum_{i=1}^{N}b_i\leqslant B\\ R_i(t)\geqslant R_0\end{cases}$	$\begin{cases}\max\sum_{i=1}^{N}m_i\\ \text{s.t.}\sum_{i=1}^{N}e_i=E\end{cases}$	$\begin{cases}\min C(E)=\sum_{i=1}^{N}C_i(e_i)\\ \text{s.t.}\sum_{i=1}^{N}e_i\leqslant E, e_i\geqslant 0\\ R_i(t)\geqslant R_0\end{cases}$

注：①e_i 为构件 C_i 被分配到的测试工作量；E 为全部可得的测试工作量；b_i、B 分别为 C_i 花费的和全部可得的预算（费用）；②$a(t)$ 为截至到 t 时软件中的累积总故障个数；$m(t)$ 为截至到 t 时累积被检测到的故障个数；m_i 为构件 C_i 被检测到的故障数量；③R_0 为预期的最低可靠性值；$R(t)$ 为在 t 时软件可靠性值；$R_i(t)$ 为 t 时构件 C_i 的可靠性值；④$C(E)$ 为整个软件所花费的成本是消耗的测试工作量 E 的函数；$C_i(e_i)$ 则对应构件 C_i 的情况.

2.3　成本模型

从软件生命周期 T_{LC} 的角度，软件成本 $E[C(T)]$ 由测试阶段成本 $C_{\text{Test}}^{t<T}$ 与操作运行阶段成本 $C_{\text{Operation}}^{t\geqslant T}$ 两大部分构成. 不同成本模型的差异主要体现在：1）测试与运行阶段所包含的子成本定义不同；2）$m(t)$ 与 $W(t)$ 的不同. 软件成本在构成上，可分为传统成本与担保及风险成本[1, 20]，如表 2 所示，其中，T 为发布时间.

表 2　软件成本函数构成

成本构成	基本含义
C_0 ①	测试初始成本，现有研究中多是自行设定初值[1, 18-19,21].
$C_1m(T)$ ①	测试阶段修复故障的成本[16,18-19,22].
$C_2f(T)$ ①	测试阶段测试资源所花费的成本，其中 $f(t)$ 为测试资源花费，有以下几种形式：$W(T)=\int_0^T w(t)\,\mathrm{d}t$[18-19]，其中：$W(t)$ 为测试工作量函数；$w(t)$ 为测试工作量的消耗率函数；T[16,20,22-24]；T^{α}[1, 25]；$\int_0^T \mathrm{e}^{-\alpha t}\,\mathrm{d}t$[21]；$m(T)$[1].
$C_3(m(T_{LC})-m(T))$ ①	操作运行阶段修复故障的成本. 其中，T_{LC} 为软件生命周期，一般认为是 ∞[16,22] 或自行设定[18-19]，但要确保 $m(T_{LC})\to m(\infty)$.
$C_4(1-R(x\mid T))$ ①	由于软件失效所导致的成本[1, 20,26]，也称为风险成本.
$C_5\int_T^{T+T_w}\lambda(t)\mathrm{e}^{-\alpha t}\,\mathrm{d}t$ ②	担保成本[21,27] 是软件发布后的担保周期内为维护软件正常运行所花费的成本；文献[1, 20] 中认为担保成本就是操作运行阶段修复故障的成本，即 $T_{LC}=T_w$（担保周期）.
$C_6(T)=\begin{cases}(T-T_d)C_p(T-T_d), & T\geqslant T_d\\ 0, & \text{otherwise}\end{cases}$ ②	惩罚成本[28-29] 是由软件延期发布或交付引发的成本. C_p 为惩罚成本函数；T_d 为软件预定发布时间.

注：①传统成本，在现有大多数研究中已被采用[1, 16,18-20,22-25]；②特殊成本，在现有研究中应用较少[1, 20-21].

当考虑传统成本时，软件成本函数 $E[C(T)]$ 为

$$\begin{aligned}E[C(T,p,r)]&=C_{\text{Test}}^{t<T}+C_{\text{Operation}}^{t\geqslant T}=\\&[C_0+C_1m(T)+C_2f(T)]+\\&[C_3(m(T_{LC})-m(T))+\\&C_4(1-R(x\mid T))].\end{aligned}$$

图 3 给出了从测试准备（$t=0$）到整个生命周期结束（$t=T_{LC}$）过程中所涉及到的成本构成.

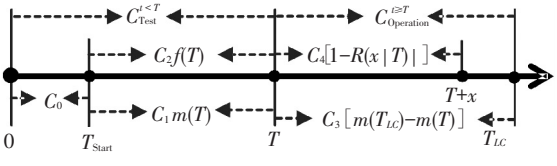

图 3　软件成本构成

图 3 中 C_1、C_3 分别为测试与操作运行阶段修复一

个故障的成本,且 $C_3 > C_1$[15,22,28,30];C_2 为单位时间测试成本;C_4 为风险成本,即因软件失效导致的成本.此外,软件发布后的操作运行阶段的可靠性 $R(x \mid T)$ 与测试阶段并不相同,因此要区别对待,如

$$R(x \mid t) = \begin{cases} e^{-(m(t+x)-m(t))}, & t \geqslant 0, x > 0, \text{Test Stage}; \\ e^{-\lambda(t)x} = e^{-\left(\frac{dm(t)}{dt}\right)x}, & t \geqslant 0, x > 0, \text{Operation Stage}. \end{cases}$$

2.4 最优发布策略

软件经过必要的测试之后,达到需求规定的目标后,可进行发布或交付给用户.一般而言,最优发布与成本紧密相关[1,16,18-20,22-25]. Okumoto 等[31]最早从成本——获益的角度研究了软件最优发布策略,之后的研究中最优发布问题在以下 4 种情况下被深入研究,如表 3 所示.

表 3 软件最优发布研究情况比较

① 最小化期望成本值[18,22,28,32]	② 在可靠性受限下最小化期望成本值[1,19,24,28,33-34]	③ 在期望成本受限下最大化可靠性[35]	④ 在剩余故障受限下最小化期望成本值[28]
$\min E[C(T)]$ or $\begin{cases}\min E[C(T)] \\ \text{s.t. } E[C(T)] \leqslant C_{\text{Constraint}}\end{cases}$	$\begin{cases}\min E[C(T)] \\ \text{s.t. } R(x \mid T) \geqslant R_0\end{cases}$	$\begin{cases}\max R(x \mid T) \\ \text{s.t. } E[C(T)] \leqslant C_0\end{cases}$	$\begin{cases}\min E[C(T)] \\ \text{s.t. } m(T) \leqslant F_{\text{remaining}}\end{cases}$

其中前两种情况较后两种要广泛得多,但随着测试目标观念的转变以及测试资源管控水平的提高,第 3 种最优发布已开始被逐渐接受.这里,需要指出:

1) 以上给出的均是成本的期望值 EC: $E[C(T)]$, Yang Bo 等[36]依据概率论(大数定律)指出,即便 $E[C(T)]$ 已达到最小值,很可能出现实际成本值 $AC > E[C(T)]$,造成成本超支.可计算概率 $P_1(T) \equiv Pr\{C(T) > E[C(T)]\}$ 来进行此种不确定性的风险分析.

2) 在求解过程中,对于 SRGM 中 $m(t)$ 内的参数应该是基于实际的测试数据估计得到[16],而并非是单纯的借助解析或数值方法[22]来求得.

3) $E[C(T)]$ 可能会因涉及 $m(t)$ 和 TE 而非常复杂,运用解析方法求解较为困难时宜采用非线性规划软件(AMPL)[13]以及数值求解软件(MatheMatica, Matlab, SPSS, 1stOpt)来运算.

3 关键技术问题讨论

3.1 软件体系结构建模

相比于忽略系统的具体结构,认为所有构件地位相当[2,3,5,9,11,13,15],近来的测试资源/时间分配研究充分地考虑到了软件内部的体系结构[2,8,9,14,37-38].文献[8]提出了考虑多因素的 TE 分配优化框架,兼顾到单一构件 C_i 对 CBS 可靠性 R 的贡献,决定为其分配合适的测试工作量 TE,用以实现预期的最小测试工作量花销下的 CBS 可靠性目标 $R_{\text{objective}}$.构件 C_i 对 CBS 可靠性 R 的贡献取决于两个因素:CBS 系统体系结构和 C_i 的成本——可靠性关系.体系结构建模是对软件内部结构特征和可能运行行为的一种描述.

1) 系统级别的建模.系统级别的建模依据构件的可靠性 R_i 与系统结构 A 来进行.对于一个终止应用来说,其软件结构可通过一个吸收的离散时间马尔科夫链一步转移概率矩阵 $\boldsymbol{P}_{n\times n}$ 来表示.这些概率可通过操作剖面来获得,操作剖面表示软件实际的使用情况为

$$A = f(\boldsymbol{P}_{n\times n}) = f(\text{Operational Profile}).$$

2) 构件级别的建模. 构件 C_i 建模表示为花销的 TE 与该构件 C_i 的可靠性 R_i 之间的关系.影响构件 C_i 可靠性 R_i 的因素之一是测试中 C_i 消耗的 TE.即便所有构件花费相同的 TE,所获得构件可靠性也不尽相同.C_i 消耗的 TE 与所获得的可靠性 R_i 之间的定量关系可通过 SRGM 和 LLP (log linear process function)来表示.

总体而言,当前研究将软件体系结构建模为各类马尔科夫模型居多,使得可以用随机过程的理论来处理;但马尔科夫模型的假设较为苛刻,而真实的软件运行具有很强的不确定性,这使得建模偏离实际较大,因而后续研究中需要考虑更多的实际情况来建模.

3.2 软件测试过程建模方法

软件测试过程中,结合具体的 SRGM 进行测试资源的管控与成本分析及最优发布,占有较大比例[2,3,6-7,9,12,15,17,39],这其中又有部分研究明确提及到 TE[2-3,18-19].SRGM 的核心是确定合适的累积的故障检测数量函数 $m(t)$[20,25]. 由于不同研究对测试环境与过程在认识上的差异,已有大量的 SRGMs 被提出.本文研究中,这些 SRGMs 可以分为完美/不完美排错、考虑/不考虑 TE,表 4

中给出了部分 SRGM 比较.显然,涵盖不完美排错与 TE 的 SRGM 能够更加准确地描述实际的测试过程,对资源分配与控制更加精细,所建立的成本更为全面,但无疑会增加求解和分析的难度.

3.3　非线性最优化求解与仿真求解

在测试资源管控与最优发布的求解中,待求解的表达式可归结为两大类解决方法:1)拉格朗日乘子法(含库恩-塔卡条件(KTC))、鞍值问题、GA、动态规划 DP 等(非线性)最优化方法;2)在解析或数值解法难以处理复杂的公式化问题时,仿真[5,36,40]可被作为优选.

3.3.1　拉格朗日乘子法(LMM)及库恩-塔卡条件(KTC)

LMM 在测试资源分配模型中已被广泛应用[3,6-7,9,11-13,15,39],其本质是把约束条件乘以拉格朗日乘子 λ 后加到待求目标函数上求极值的方法,即构造拉格朗日等式为

$$\mathrm{LME} = \lambda \cdot [\text{Constraint Condition}] + [\text{Objective Function}].$$

同时,KTC 则给出了求极大/小值的必要性条件.在最小化剩余故障进行测试资源分配的研究中,文献[3]构造了拉格朗日等式来求解资源分配结果 X_i 为

$$\min: L(X_1,X_2,\cdots,X_N,\lambda) = \lambda \cdot \Big[\sum_{i=1}^{N} X_i - W + \sum_{i=1}^{N} \overline{R_i}\Big] + \Big[\sum_{i=1}^{N} v_i a_i \mathrm{e}^{-r_i R_i} \mathrm{e}^{-r_i X_i}\Big]. \tag{1}$$

基于 KTC,式(1)存在最小值的必要条件为

$$\begin{cases} A_1: \dfrac{\partial L(X_1,X_2,\cdots,X_N,\lambda)}{\partial X_i} \geqslant 0,\ i = 1,2,\cdots,N; \\ A_2: X_i \dfrac{\partial L(X_1,X_2,\cdots,X_N,\lambda)}{\partial X_i} = 0,\ i = 1,2,\cdots,N; \\ A_3: \lambda \cdot \Big\{\sum\limits_{i=1}^{N} X_i - \Big(W - \sum\limits_{i=1}^{N} \overline{R_i}\Big)\Big\} = 0,\ i = 1,2,\cdots,N. \end{cases}$$

一般而言,这类问题通常可归结为:SRGM 结合 LMM 等最优化方法求解模式:

1) 若出现负解(即某 C_i 被分配的 $e_i < 0$),则需要执行改进算法进行优化[3,6,13,15,17].为防止负解出现,在所建立的资源分配优化模型中加上必要的限制条件即可.

2) 若出现零解(即某 C_i 被分配的 $e_i = 0$),这也是不允许的.因为这表明此时 C_i 实际上没有被分配测试资源,无法被测试[11].其解决的办法是重新设定资源分配条件,但这会增加求解的难度.

3)若采用 LMM(以及 KTC)不能提供封闭形式的解时,DP 中的 Bellman 优化规则方法可成为一种选择.

3.3.2　离散事件仿真(discrete event simulation, DES)

DES 是基于率函数控制的事件过程仿真方法[41].在无穷小的时间间隔 $(t, t + \mathrm{d}t)$ 内,事件 A 发生的条件概率为 $\lambda(t)\mathrm{d}t$,其中 $\lambda(t)$ 即为率函数,表示 t 时刻单位时间内故障发生的个数,即 $\lambda(t) = \dfrac{\mathrm{d}m(t)}{\mathrm{d}t}$;而事件 A 可以用来表示软件失效故障检测计数随机过程 $\{X(t)\}$.DES 通过随机产生 $[0,1]$ 内的随机数 x,并判断 $x < \lambda(t)\mathrm{d}t$,若成立,则认为故障被检测到.此外,文献[36,40]中应用蒙特卡罗仿真(MCS)来进行复杂概率事件的近似估计.仿真方法的优势在于能够放宽假设条件,使其能够处理数值解法不能胜任的情况;其不足是仿真只能给出一种趋势性的结论,很难做出精确的判断.

4　典型案例模型研究比较

在上述分析的基础上,表 4 选择 8 个典型的模型从 5 个方面进行归一化归类分析,对测试资源分配与成本管控和最优发布技术问题进行了比较.

从表 4 可以看出,对于单一形式的软件,研究主要侧重于成本管控和最优发布,主要基于指数型 SRGM 所建立的简洁的公式化形式,并以解析求解方法为主;由于构件软件考虑到内部的体系结构特征,对测试资源的分配与管控和成本模型较为关注,所建立的公式化描述要复杂些,这使得求解方法更为繁琐.测试资源分配与成本管控和最优发布技术问题可被建模为特定目标下的最优化问题,且正在向多目标融合方向发展.另外,由于模型中所考虑的因素也越来越丰富,传统解析方法难以应对,相应的非线性求解方法逐渐成为优选.

总体而言,测试资源与成本管控和最优发布尚需要在以下几个方面解决所面临的问题.

1)测试资源的分配与管控应考虑到软件体系结构特征,填补可靠性需求与资源平衡配置之间的差距.

2)由于实际随机因素的不确定,未来研究中应侧重考虑更多不完美排错相关的 SRGM 来建立各类优化模型.

3)资源分配与控制的目的是实现成本模型受控下最优发布软件.

4)测试资源与成本管控和最优发布问题应与可靠性的评估紧密联系起来,实现以可靠性为核心的研究重点.

表 4 研究内容解析与典型模型比较

模型	类型①	对象/阶段②	SRGM③	问题公式化描述	方法④
Huang-II model[19]	C+D	SS	EX+PD+TE	$\begin{cases}\min E[C(T)]\\ \text{s.t. } R(x \mid T) \geqslant R_0\end{cases}$	AM
Kapur-II model[16]	C+D	SS	EX+ID	$\begin{cases}\min C(T,p,r),\ 0<r<p<1\\ \text{s.t. } R(x \mid T) \geqslant R_0\end{cases}$ p: probability of complete debugging r: probability of introducing new faults	AM
Kapur-III model[33]	C+D	SS	–	$\begin{cases}\min E[C(T)]\\ \text{s.t. } R(x \mid T) \geqslant R_0\end{cases}$	GA
Kapur-I model[11]	A+B	CBS+UT	EX+PD+TE	$\begin{cases}\max \sum_{i=1}^{N} m_i = \sum_{i=1}^{N} a_i(1-\mathrm{e}^{-b_i e_i})\\ \text{s.t. } \sum_{i=1}^{N} e_i = E, e_i \geqslant 0\end{cases}$	DP
Leung model[5]	A+B	CBS+UT	EX+PD+TE	$\begin{cases}\min \sum_{i=1}^{N}(a-m(T))\\ \text{s.t. } \sum_{i=1}^{N} e_i = E, e_i \geqslant 0\end{cases} + \begin{cases}\min \sum_{i=1}^{N} e_i\\ \text{s.t. } \sum_{i=1}^{N}(a-m(T)) = Z\end{cases}$	LMM+DES
Fiondella model[8]	A+B	CBS+UT	EX+PD	$\begin{cases}\min \sum_{i=1}^{n} e_i = e\\ \text{s.t. } R \geqslant R^*\\ \quad \forall i(e_i \geqslant 0) \wedge \{e_i \leqslant \inf\{e_i \in R; r_i \geqslant 1\}\}\end{cases}$	NM(GA/DP)
Jha model[2]	A+B+C	CBS+UT	(EX+SS)+PD+TE	$\begin{cases}\min C(E) = \sum_{i=1}^{N} C_i(e_i)\\ \text{s. t. } \sum_{i=1}^{N} e_i \leqslant E,\ e_i \geqslant 0, i = 1,\cdots,N\\ \quad R_i(t) = \dfrac{1-\mathrm{e}^{-b_i e_i}}{1+\beta_i \mathrm{e}^{-b_i e_i}} \geqslant R_0\end{cases}$	SVP (鞍点为最优解)
Huang-I model[15]	A+C	CBS+UT	EX+PD+TE	$\begin{cases}\min C(E) = \sum_{i=1}^{N} C_i(e_i)\\ \text{s. t. } \sum_{i=1}^{N} e_i \leqslant E,\ e_i \geqslant 0\\ \quad R_i(t) \geqslant R_0\end{cases}$	LMM(DP)

注：① 类型包括：A 为资源分配；B 为资源控制；C 为成本模型；D 为最优发布. ② 对象包括：SS 为 单一软件；CBS 为构件软件. 阶段包括：UT 为单元/模块测试；IT 为集成测试. ③ SRGM 涵盖：EX 为指数型；SS 为 S 型；PD 为完美排错；ID 为不完美排错；TE 为测试工作量. ④ 方法包括：AM 为解析方法；NM 为非线性方法；LMM 为拉格朗日乘子法；KTC 为库恩-塔卡条件；DP 为动态规划；GA 为基因算法；SVP 为鞍值问题；DES 为离散事件仿真(基于率的仿真)；MCS 为蒙特卡罗仿真.

5 结 论

1)软件测试已由过去的注重检测与排除掉故障过渡到以不断提高其可靠性为目标.因此,以可靠性为核心也成为了测试资源与成本管控和最优发布研究问题的关键.

2)执行测试资源管控为软件管理者如何高效优化定量地分配与管理测试资源,控制开发成本,决策软件最优发布提供了必要的决策支持.

3)如何管控有限的测试资源,使得在受限的开发测试时间内,在最小软件成本的约束下最优发布可靠的软件系统已成为多目标优化融合下的研究关注点.

参考文献

[1] PHAM H, ZHANG Xuemei. A software cost model with warranty and risk costs [J]. IEEE Transactions on Computers, 1999, 48(1): 71-75.

[2] JHA P C, GUPTA D, YANG Bo, et al. Optimal testing resource allocation during module testing considering cost, testing effort and reliability [J]. Computers & Industrial Engineering, 2009, 57(3): 1122-1130.

[3] HUANG C Y, LYU M R. Optimal testing resource allocation, and sensitivity analysis in software development [J]. IEEE Transactions on Reliability, 2005, 54(4): 592-603.

[4] OHTERA H, YAMADA S. Optimal allocation and control problems for software-testing resources[J]. IEEE Transactions on Reliability, 1990, 39(2): 171-176.

[5] LEUNG Y W. Dynamic resource-allocation for software-module testing [J]. Journal of Systems and Software, 1997, 37(2): 129-139.

[6] HUANG C Y, LO J H, KUO S Y, et al. Optimal allocation of testing resources for modular software systems [C]//Proceedings of 13th International Symposium on Software Reliability Engineering, Annapolis, Maryland: IEEE, 2002: 129-138.

[7] YAMADA S, ICHIMORI T, NISHIWAKI M. Optimal allocation policies for testing-resource based on a software reliability growth model[J]. Mathematical and Computer Modelling, 1995, 22(10): 295-301.

[8] FIONDELLA L, GOKHALE S S. Optimal allocation of testing effort considering software architecture[J]. IEEE Transactions on Reliability, 2012, 61(2): 580-589.

[9] XIE Min, YANG Bo. Optimal testing-time allocation for modular systems[J]. International Journal of Quality & Reliability Management, 2001, 18(8): 854-863.

[10] BERMAN O, CUTLER M. Resource allocation during tests for optimally reliable software [J]. Computers & Operations Research, 2004, 31(11): 1847-1865.

[11] KAPUR P K, JHA P C, BARDHAN A K. Dynamic programming approach to testing resource allocation problem for modular software [J]. Journal of Applied Mathematics, 2003, 14: 27-40.

[12] KHAN M G M, AHMAD N, RAFI L S. Optimal testing resource allocation for modular software based on a software reliability growth model: a dynamic programming approach[C]//International Conference on Computer Science and Software Engineering. Wuhan, Hubei: IEEE, 2008: 759-762.

[13] LYU M R, RANGARAJAN S, Van MOORSEL A P A. Optimal allocation of test resources for software reliability growth modeling in software development[J]. IEEE Transactions on Reliability, 2002, 51(2): 183-192.

[14] PIETRANTUONO R, RUSSO S, TRIVEDI K S. Software reliability and testing time allocation: an architecture-based approach[J]. IEEE Transactions on Software Engineering, 2010, 36(3): 323-337.

[15] HUANG C Y, LO J H. Optimal resource allocation for cost and reliability of modular software systems in the testing phase [J]. Journal of Systems and Software, 2006, 79(5): 653-664.

[16] KAPUR P K, GUPTA D, GUPTA A, et al. Effect of introduction of fault and imperfect debugging on release time[J]. Journal of Applied Mathematics, 2008:62-90.

[17] KUBAT P, KOCH H S. Managing test-procedures to achieve reliable software [J]. IEEE Transactions on Reliability, 1983, 32(3): 299-303.

[18] HUANG C Y, LYU M R. Optimal release time for software systems considering cost, testing-effort, and test efficiency [J]. IEEE Transactions on Reliability, 2005, 54(4): 583-591.

[19] HUANG C Y. Cost-reliability-optimal release policy for software reliability models incorporating improvements in testing efficiency[J]. Journal of Systems and Software, 2005, 77(2): 139-155.

[20] ZHANG Xuemei, PHAM H. A software cost model with error removal times and risk costs [J]. International Journal of Systems Science, 1998, 29(4): 435-442.

[21] WILLIAMS D R P. Study of the warranty cost model for software reliability with an imperfect debugging phenomenon[J]. Turk J Elec Engin, 2007, 15(3): 369-381.

[22] XIE Min, YANG Bo. A study of the effect of imperfect debugging on software development cost [J]. IEEE Transactions on Software Engineering, 2003, 29(5): 471-473.

[23] HUANG C Y, LIN C T. Analysis of software reliability modeling considering testing compression factor and failure-to-fault relationship [J]. IEEE Transactions on Computers, 2010, 59(2): 283-288.

[24] BOLAND P J, NÍ CHUÍV N. Optimal times for software release when repair is imperfect [J]. Statistics & Probability Letters, 2007, 77(12): 1176-1184.

[25] ZHANG Xuemei, PHAM H. A software cost model with warranty cost, error removal times and risk costs [J]. IIE Transactions, 1998, 30(12): 1135-1142.

[26] LIU C T, CHANG Y C. A reliability-constrained software release policy using a non-Gaussian Kalman filter model [J]. Probability in the Engineering and Informational Sciences, 2007, 21(2): 301-314.

[27] WILLIAMS D R P. Optimal release policies for a software system with warranty cost and change-point

phenomenon [J]. Turkish Journal of Electrical Engineering & Computer Sciences, 2013, 21(1): 234-245.

[28] PHAM H. A software cost model with imperfect debugging, random life cycle and penalty cost[J]. International Journal of Systems Science, 1996, 27(5): 455-463.

[29] HOU R H, KUO S Y, CHANG Y P. Optimal release times for software systems with scheduled delivery time based on the HGDM [J]. IEEE Transactions on Computers, 1997, 46(2): 216-221.

[30] BOEHM B, ABTS C, CHULANI S. Software development cost estimation approaches——A survey [J]. Annals of Software Engineering, 2000, 10(1/4): 177-205.

[31] OKUMOTO K, GOEL A L. Optimum release time for software systems based on reliability and cost criteria[J]. Journal of Systems and Software, 1980, 1: 315-318.

[32] MORALI N, SOYER R. Optimal stopping in software testing[J]. Naval Research Logistics (NRL), 2003, 50(1): 88-104.

[33] KAPUR P K, PHAM H, AGGARWAL A G, et al. Two dimensional multi-release software reliability modeling and optimal release planning[J]. IEEE Transactions on Reliability, 2012, 61(3): 758-768.

[34] JAIN M, GUPTA R. Optimal release policy of module-based software[J]. Quality Technology and Quantitative Management, 2011, 8(2): 147-165.

[35] LEUNG Y W. Optimum software release time with a given cost budget[J]. Journal of Systems and Software, 1992, 17(3): 233-242.

[36] YANG Bo, HU Huajun, JIA Lixin. A study of uncertainty in software cost and its impact on optimal software release time [J]. IEEE Transactions on Software Engineering, 2008, 34(6): 813-825.

[37] WANG Zai, TANG Ke, YAO Xin. Multi-objective approaches to optimal testing resource allocation in modular software systems[J]. IEEE Transactions on Reliability, 2010, 59(3): 563-575.

[38] KAPUR P K, AGGARWAL A G, KAUR G. Simultaneous allocation of testing time and resources for a modular software[J]. International Journal of System Assurance Engineering and Management, 2010, 1(4): 351-361.

[39] HOU R H, KUO S Y, CHANG Y P. Needed resources for software module test, using the hyper-geometric software reliability growth model[J]. IEEE Transactions on Reliability, 1996, 45(4): 541-549.

[40] MASUDA Y, MIYAWAKI N, SUMITA U, et al. A statistical approach for determining release time of software system with modular structure [J]. IEEE Transactions on Reliability, 1989, 38(3): 365-372.

[41] HUANG R, LYU M R, Kanoun K. Simulation techniques for component-based software reliability modeling with project application [C]//Proceedings of International Symposium on Information Systems and Engineering. Hong Kong: Comp Sci Res, Educ, & Applicat Press, 2001: 283-289.

(编辑 张 红)